Verfahren zur computerunterstützten Inhaltsanalyse in der Kommunikationswissenschaft

Abhandlung (kumulative Dissertation)

zur Erlangung der Doktorwürde

der Philosophischen Fakultät

der

Universität Zürich

vorgelegt von

Martin Wettstein

Angenommen im Herbstsemester 2015

auf Antrag der Promotionskommission:

Prof. Dr. Werner Wirth (hauptverantwortliche Betreuungsperson)

Prof. Dr. Carsten Wünsch

Zürich, 2016

Inhaltsverzeichnis

Kapitel 1. Forschungsinteresse ... 2

Kapitel 2. Inhaltsanalyse als Methode in der Sozialwissenschaft ... 4

2.1. Inhaltsanalyse als Methode in der Kommunikationswissenschaft ... 8
2.2. Schematischer Ablauf von Inhaltsanalysen ... 11
2.2.1. Planungsphase ... 12
2.2.2. Entwicklungsphase ... 15
2.2.3. Testphase ... 18
2.2.4. Anwendungsphase ... 20

Kapitel 3. Computerunterstützte Inhaltsanalyse ... 23

Kapitel 4. Computerunterstützte Planungsphase ... 26

4.1. Textauswahl ... 26
4.1.1. Term-Mapping zur Bestimmung der Unterthemen ... 27
4.1.2. Longitudinale Analyse zur Bestimmung von Kommunikationsereignissen ... 31
4.1.3. Optimieren der Auswahl von Schlüsselworten zur Textakquise ... 33
4.1.4. Anwendungsbeispiel: Debatte zur Arbeitsmarktpolitik ... 37
4.2. Projektplanung ... 42
4.2.1. Datenbanklösung für ein Grossprojekt ... 43

Kapitel 5. Computerunterstützte Entwicklungsphase ... 47

5.1. Textakquise ... 47
5.1.1. Zugänge ... 47
5.1.2. Maschinelles Lernen zur Auswahl relevanter Texte ... 48
5.1.3. Vorbereitung des Textkorpus ... 51
5.2. Erhebungsinstrument ... 51
5.2.1. Vorbereitung paralleler computergestützter Inhaltsanalyse ... 52
5.2.2. Vorbereitung halbautomatischer Inhaltsanalyse ... 53

Kapitel 6. Computerunterstützte Testphase ... 55

6.1. Eignungs-, Reliabilitäts- und Validitätstests ... 55

Kapitel 7. Computerunterstützte Anwendungsphase ... 69

7.1. Erhebung ... 69
7.1.1. Dateneingabe über eine Eingabemaske ... 70
7.1.2. Halbautomatische Erfassung von Texten ... 72
7.2. Datenaufbereitung ... 75
7.2.1. Identifikation von Mustern in Inhaltsanalysedaten ... 75
7.2.2. Verlaufsanalyse zur Erforschung von Aufmerksamkeitsschüben ... 76
7.3. Betreuung ... 78
7.3.1. Automatische Zuweisung von Texten und Sammlung von Daten ... 78
7.3.2. Messung von Randdaten der Erhebung ... 80
7.3.3. Erfassung möglicher schadhafter Einflüsse ... 80

Kapitel 8. Zusammenfassung ... 82

8.1. Der Prozess der computerunterstützten Inhaltsanalyse ... 82
8.2. Beitrag dieser Dissertation ... 85
8.3. Lücken im Methodeninventar ... 86

Kapitel 9. Fazit ... 87

Literatur ... 88

Anhänge ... 94

Kapitel 1. Forschungsinteresse

In unterschiedlichen Forschungszweigen der empirischen Sozialwissenschaft ist die Analyse von Texten wie Medieninhalten, Pressemitteilungen, Büchern oder Werbung erforderlich, um zentrale Forschungsfragen zu beantworten. Traditionell werden diese Inhaltsanalysen durch geschulte Codierer vorgenommen, welche die Texte nach klaren Vorgaben analysieren, abstrahieren und in eine Form übersetzen, die sich für die Beantwortung von Forschungsfragen oder für den Test von Hypothesen eignet.

Während sich die manuelle Inhaltsanalyse als Methode in der empirischen Sozialwissenschaft etabliert hat und die zentrale Erhebungsmethode der Kommunikationswissenschaft ist (vgl. Wirth & Lauf, 2001: 7; Lombard, Snyder-Duch & Bracken, 2002: 587), kann sie mitunter auch sehr zeit- und kostenintensiv sein. Für Forschungsprojekte, für die eine grosse Anzahl von Texten aus mehreren Quellen relevant ist, steigt daher der Druck auf Forscher, auf die effizienteren automatischen Verfahren zur Textanalyse auszuweichen und auf eine manuelle Analyse zu verzichten.

Gerade in der Kommunikationswissenschaft bestehen jedoch noch heute begründete Vorbehalte gegen den Ersatz menschlicher Codierer durch einen Computer, wenn es darum geht, mehr als nur das Vorhandensein von Schlüsselbegriffen oder Namen in Texten zu identifizieren. Bei der inhaltlichen Analyse von Interpretationen oder journalistischer Aufbereitung wird menschlichen Codieren eine gewisse Abstraktionsfähigkeit und ein umfassendes Vorwissen zum Thema abverlangt. Dazu zählt besonders bei der Analyse journalistischer Texte die Fähigkeit, bildhafte Sprache, Ironie und Sarkasmus, neue Spitznamen für Politiker und Rückbezüge auf themenverwandte Ereignisse aus dem Kontext zu erkennen und zu verstehen.

Computerprogramme können diese hohen Anforderungen, die an menschliche Codierer bei der Analyse von Medienberichterstattung gestellt werden, oftmals nicht erfüllen und die Validität automatischer Codierentscheidungen wird bei komplexen inhaltlichen Kategorien in Zweifel gezogen. In der Konsequenz werden Computer in der Kommunikationswissenschaft noch heute kaum für inhaltliche Analysen eingesetzt, sondern vornehmlich für die Textakquise und die Datenspeicherung und –Aufbereitung während einer Inhaltsanalyse. Für diese Aufgaben haben sie sich als verlässliche und effiziente Hilfsmittel erwiesen (vgl. Stuckardt, 2000: 25f).

Zusätzlich zu diesen traditionellen Aufgaben von Computerprogrammen in der Inhaltsanalyse beschreibt Luzar (2004) auch die Möglichkeit, Daten direkt am Bildschirm über eine Eingabemaske einzugeben, als sinnvolles Einsatzgebiet von Computern. Scharkow (2012: 41) treibt die Idee weiter und stellt die Möglichkeit zur Diskussion, auch an anderen Stellen im Prozess einer Inhaltsanalyse automatische Verfahren einzusetzen. Während die eigentliche Erhebung weiterhin durch einen

menschlichen Codierer geleistet wird, könnten Computer an verschiedenen anderen Stellen einer Inhaltsanalyse eingesetzt werden, um die Arbeit der Forscher und der Codierer massgeblich zu erleichtern. Der Aufwand einer Inhaltsanalyse kann damit reduziert werden, ohne die Validität der Ergebnisse zu gefährden und auf menschliches Textverständnis zu verzichten.

In dieser Dissertation wird der Ansatz einer computerunterstützten Inhaltsanalyse systematisch ausgebaut und ausformuliert. Das Ziel ist es, ein umfangreiches Methodeninventar für eine computerunterstützte Inhaltsanalyse zu skizzieren, das vollständig oder teilweise eingesetzt werden kann, um den Prozess einer manuellen Inhaltsanalyse effizienter zu gestalten. Das resultierende Methodeninventar setzt sich zu einem grossen Teil aus Verfahren zusammen, die im Rahmen dieser Dissertation für zwei umfangreiche internationale Inhaltsanalysen öffentlicher Debatten entwickelt wurden. Diese Grossprojekte boten auch jeweils Gelegenheit, die Verfahren unter realen Bedingungen zu testen. Den Kern der Dissertation bilden vier Verfahren, die für eine Publikation vorbereitet oder bereits publiziert wurden. Einige weitere Verfahren werde ich ausschliesslich in dieser Rahmenschrift vorstellen und diskutieren. Um das Methodeninventar zu vervollständigen, werde ich an Stellen, an denen bereits durch andere Forscher Verfahren entwickelt wurden, auf diese hinweisen.

Formal setzt sich die Dissertation aus drei Teilen zusammen: Der erste Teil besteht aus vier unabhängigen Publikationen in Alleinautorenschaft, in denen ich automatische Verfahren zur Vorbereitung, Erhebung und Datenaufbereitung vorstelle. Der zweite Teil ist diese Rahmenschrift, welche die Publikationen systematisch in den Prozess der Inhaltsanalyse einordnet und sie durch weitere, unveröffentlichte und von anderen Forschern stammende Verfahren ergänzt. Der dritte Teil ist eine Softwarelösung für die konkrete Durchführung der computerunterstützten Inhaltsanalyse und der einzelnen hier vorgestellten Verfahren. Sie besteht aus einer Eingabemaske für Daten (*Angrist*), einem Programm zur Datenaufbereitung und -Auswertung (*Nogrod*) und einem Programm zur automatischen Textanalyse (*Aeglos*) (Siehe Anhang B). Die Arbeit nimmt damit insgesamt eine vorwiegend pragmatische Perspektive ein und ist nicht auf die Theorie, sondern auf das Forschungsprogramm von Inhaltsanalysen fokussiert.

Kapitel 2. Inhaltsanalyse als Methode in der Sozialwissenschaft

Inhalts-, Diskurs- oder Dokumentenanalysen sind die zentrale Erhebungsmethode der Kommunikationswissenschaft und werden auch in anderen Sozialwissenschaften zunehmend eingesetzt, wenn es darum geht, Inhalte von Massenkommunikation, Organisationskommunikation oder interpersonaler Kommunikation zu untersuchen. Dabei unterscheiden sich je nach Anwendungsgebiet nicht nur die Erhebungsmethoden, sondern auch die Definitionen der Methode der Inhaltsanalyse teilweise erheblich (vgl. Mayring, 2007: 11f).

Für diese Arbeit wird die Methode der Inhaltsanalyse so allgemein und inklusiv wie möglich definiert, um nicht nur unterschiedliche qualitative und quantitative Verfahren, sondern auch die automatische Inhaltsanalyse einzuschliessen. Eine solche inklusive Definition findet sich bei Früh (2009), der drei zentrale Definitionsbestandteile herausarbeitet: Erstens ist die Inhaltsanalyse eine empirische Methode, deren Zugang zur Realität über Erfahrung, Beobachtung und Messung führt. Sie ist dabei aber nicht als Erhebungs- oder Messmethode zu verstehen, sondern als Forschungsmethode, die unterschiedliche Erhebungs- und Messmethoden einschliessen kann (S. 27-28). Zweitens bedingt eine wissenschaftliche Inhaltsanalyse ein systematisches Vorgehen und beinhaltet neben der Messung auch die Auswahl von Untersuchungsmaterial und die Entwicklung geeigneter Messinstrumente für eine systematische Untersuchung (S. 40). Drittens soll die Methode intersubjektiv nachvollziehbar sein, was nur über die Offenlegung der Messung und eine Dokumentation und Auswertung der Reliabilität der Messung gewährleistet werden kann (S. 40).

Die Inhaltsanalyse wird mit dieser allgemeinen Definition zu einem Forschungsprogramm oder einer "Forschungsperspektive" (Wedl, Herschinger & Gasteiger, 2014: 538), die mehrere Arbeitsschritte und unterschiedliche Erhebungsmethoden beinhaltet, um aus Texten relevante Informationen zu extrahieren und aufzubereiten, die für die Beantwortung einer Forschungsfrage erforderlich sind. Als Text kann in diesem Zusammenhang nicht nur das geschriebene Wort, sondern auch ein Bild, Videomaterial, ein Plakat, oder eine Diskussion gelten. Die Informationen, die aus diesen Texten extrahiert werden, können Themen, Argumentationen, Stilelemente, Gestaltungsmerkmale, Positionen und viele weitere inhaltliche und formale Elemente sein. Die Messung an sich ist im Rahmen einer Inhaltsanalyse zwar ein zentraler Bestandteil, macht aber nicht die gesamte Inhaltsanalyse aus. In dieser Dissertation werden deswegen die Begriffe der "Inhaltsanalyse" und der "Erhebung" strikt getrennt. Während der Begriff der Inhaltsanalyse das gesamte Forschungsprogramm von der Konzeption bis zur Auswertung umfasst, beschränkt sich die Erhebung auf die tatsächliche Messung oder Datenerhebung.

Eine Inhaltsanalyse kann also die mediale Präsentation eines Politikers untersuchen und dafür die wertenden Aussagen zu diesem Politiker auf den Titelseiten von fünf Zeitungen über den Zeitraum eines Jahres erheben und auswerten. Oder sie kann über eine Erhebung der Argumentations- und Konfliktlinien in den Beiträgen zu einer internationalen Klimakonferenz den Diskurs zu diesem Thema untersuchen und mit Parlamentsentscheidungen abgleichen. Selbst Untersuchungen zu einem Wandel journalistischer Normen und Prozesse können durch systematische Inhaltsanalysen von Medieninhalten profitieren. Schliesslich ist es auch möglich, Ergebnisse von Inhaltsanalysen mit anderen Beobachtungen zu verbinden, um beispielsweise eine Veränderung in der öffentlichen Meinung durch die Anzahl und Intensität von Argumenten in Tageszeitungen zu erklären oder die mediale Darstellung eines Politikers auf die Inhalte seiner Pressemitteilungen zurückzuführen.

Die allgemeine Definition der Inhaltsanalyse als Forschungsprogramm zur systematischen Extraktion von Informationen aus Texten lässt die Art der Erhebung grundlegend offen. Die einzige Anforderung an Erhebungsmethoden in der Inhaltsanalyse sind deren Systematik, Nachvollziehbarkeit und Dokumentation. Für die Erhebungsmethoden kann man grob zwischen Herangehensweisen unterschieden werden. Bei *quantitativen* Ansätzen werden Inhalte durch Zählung und Klassifikation erhoben. Bei *qualitativen* Ansätzen steht die Interpretation, Zusammenfassung und Kategorienbildung im Vordergrund. Für beide Herangehensweisen können sowohl manuelle als auch automatische Erhebungsmethoden eingesetzt werden.

In diesem Kapitel wird ein allgemeiner Überblick über die einzelnen Herangehensweisen der Inhaltsanalyse in der Kommunikationswissenschaft geschaffen. Anhand eines schematischen Prozesses der Inhaltsanalyse wird dabei aufgezeigt, welche Auswirkungen die Entscheidung für bestimmte Ansätze und Verfahren auf den Arbeitsablauf und den personellen und zeitlichen Aufwand einer Inhaltsanalyse haben können. Gleichzeitig wird mit der Skizzierung des schematischen Prozesses die Grundlage für die systematische Einordnung automatischer Verfahren für die Durchführung einer computerunterstützten Inhaltsanalyse geschaffen.

Quantitative Verfahren

Quantitative oder standardisierte Inhaltsanalysen haben das Ziel, eine Menge von Texten so weit zu abstrahieren, dass sie in einer numerischen Datenmatrix dargestellt und mittels statistischer Verfahren ausgewertet werden können (vgl. Rössler 2010: 20). Die Erhebungsmethoden dieser Herangehensweise beinhalten dabei die Zählung, Identifikation und Klassifikation inhaltlicher oder formaler Elemente und die Darstellung der Ergebnisse in Form einer Tabelle (vgl. Mayring, 2007: 13f).

Zählbare Elemente können die Worte in einem Text, die Anzahl Abschnitte und Bilder, die Anzahl unterschiedlicher Akteure, die Anzahl von Aussagen zu einem Thema oder die Anzahl von Personen auf einem Bild sein. Im weiteren Sinn gehören zu den zu den zählbaren Elementen auch die Dauer

eines Beitrags oder das Verhältnis zwischen Bild und Text auf einer Seite. Das Ergebnis einer Erhebung durch Zählung ist jeweils eine metrische Variable.

Bei der Identifikation von Textmerkmalen kann auf formaler Ebene das Vorkommen von Tabellen, Grafiken und direkten Zitaten erfasst werden. Inhaltlich können dagegen Fallbeispiele, überraschende Elemente oder journalistische Interpretation von zitierten Aussagen in einem Text identifiziert werden. Hier ist das Ergebnis der Erhebung jeweils eine dichotome Variable, welche angibt, ob ein Text oder eine Analyseeinheit ein bestimmtes Element enthält oder nicht.

Bei der Klassifikation semantischer Elemente geht es schliesslich um eine möglichst präzise Bestimmung einzelner Akteure, Themen, Argumentationslinien oder Frames anhand einer vordefinierten Liste oder eines Kategoriensystems. Je nach Kategorie kann das Ergebnis dieser Erhebung in Form einer ordinalen (z.B: regionale, nationale, internationale oder globale Reichweite) oder einer nominalen Variable (z.B: Politik, Wirtschaft, Sport oder Kultur Ressort) codiert werden.

Die Erhebung der Daten erfolgt bei quantitativen Inhaltsanalysen nach einem festgelegten Kategoriensystem, welches Codierregeln für jede Textkategorie enthält (vgl. Rössler, 2010: 95). Diese Regeln werden von Codierern auf die einzelnen Texte angewandt, um ihre Inhalte zu codieren, also in eine Reihe von Codes und Zähldaten zu übersetzen. Die numerische Datenmatrix, die durch eine solche Codierung mehrerer Texte entsteht, kann schliesslich aufbereitet und statistisch ausgewertet werden, um Annahmen über die Textinhalte zu prüfen oder sie mit Daten aus anderen Quellen in Zusammenhang zu bringen.

Qualitative Verfahren

Bei der qualitativen oder interpretierenden Inhaltsanalyse wird bei der Erhebung von Texten keine Datenmatrix erstellt, sondern eine inhaltliche Zusammenfassung oder Kategorisierung, welche jeden Text in seinem Erscheinungskontext interpretiert (vgl. Mayring, 2007: 42f). Die Erhebung wird dabei nicht von einem Codebuch mit konkreten Codierregeln diktiert, sondern durch Codierleitfäden gelenkt, in denen das Vorgehen bei der Analyse und der Ergebnispräsentation vorgegeben wird.

Mayring (2007) beschreibt mit den Erhebungsmethoden der Zusammenfassung, Kategorienbildung, Explikation und Strukturierung unterschiedliche Verfahren der qualitativen Inhaltsanalyse (S. 59ff). Das Resultat dieser Erhebungen ist jeweils entweder eine umfangreiche Interpretation von Texten, welche sie mit Rücksicht auf ihren Entstehungszusammenhang beschreibt und erklärt oder eine strukturierte Typologie oder Zusammenfassung einzelner Themen, Argumentationen oder Konfliktlinien in einem Diskurs.

Die Ergebnisse qualitativer Inhaltsanalysen sind eng mit den untersuchten Texten verknüpft und an Textbeispiele gebunden. Eine tabellarische Darstellung oder statistische Auswertung der Ergebnisse ist deswegen weder möglich noch für die Weiterarbeit erforderlich. Das Abstraktionsniveau dieser Analysen ist weniger hoch als bei quantitativen Analysen, dafür wird die ursprüngliche Information weitestgehend erhalten.

Automatische Verfahren

Sowohl qualitative als auch quantitative Inhaltsanalysen können teilweise durch automatische Verfahren zur Textanalyse ersetzt oder ergänzt werden. Angesichts der terminologischen Vielfalt in der Benennung automatischer, computergestützter und computerunterstützter Inhaltsanalysen bietet sich an dieser Stelle eine Begriffsdefinition für die vorliegende Arbeit an.

Eine für die Kommunikationswissenschaft sinnvolle Kategorisierung leistete Stuckardt (2000) mit der klaren Trennung zwischen *computergestützten* Inhaltsanalysen, bei denen die Codierentscheidungen durch automatische Verfahren der Textanalyse getroffen werden, und *computerunterstützten* Inhaltsanalysen, wo automatische Verfahren im Prozess der Inhaltsanalyse eingesetzt, die Codierentscheidungen aber von menschlichen Codierern getroffen werden (S. 25ff). Luzar (2004) griff diese Unterscheidung wieder auf, um mögliche Einsatzgebiete automatischer Verfahren in qualitativen und quantitativen Inhaltsanalysen zu beschreiben, welche die eigentliche Erhebung der Texte nicht betreffen. Dazu gehören unter anderem die Textakquise, die Dateneingabe, die Verwaltung von Notizen und Kategorien in qualitativen Inhaltsanalysen oder die Datenverwaltung und –aufbereitung in quantitativen Inhaltsanalysen (S. 147f). In dieser Arbeit wird an dieser Unterscheidung festgehalten und es wird zudem der von Scharkow (2012) gemachte Vorschlag beherzigt, die computergestützte Inhaltsanalyse zwecks begrifflicher Eindeutigkeit als automatische Inhaltsanalyse zu bezeichnen.

Folglich werden in dieser Arbeit jene Inhaltsanalysen als *automatische* Inhaltsanalyse bezeichnet, in denen die Codierentscheidungen nicht von menschlichen Codierern, sondern von automatischen Verfahren zur Textanalyse getroffen werden. *Manuelle* Inhaltsanalysen, auf der anderen Seite, zeichnen sich dadurch aus, dass Codierentscheidungen einem Menschen überlassen werden. Computerunterstützte Inhaltsanalysen sind eine Unterkategorie manueller Inhaltsanalysen, bei denen automatische Verfahren in einzelne Arbeitsschritte integriert werden. Die am weitesten verbreitete Form computerunterstützter Inhaltsanalyse ist eine manuelle Inhaltsanalyse, bei der die Texte mittels Volltextsuche aus digitalen Archiven beschafft werden (vgl. Stuckardt, 2000: 25).

2.1. Inhaltsanalyse als Methode in der Kommunikationswissenschaft

Dass die Inhaltsanalyse auch heute noch die zentrale Methode dieses Fachs ist, wird regelmässig betont (z.B: Wirth & Lauf, 2001: 7; Lombard, Snyder-Duch, & Bracken, 2002; Früh, 2009: 12-13). So finden auch Früh und Früh (2015) in einem systematischen Review inhaltsanalytischer Studien in mehreren sozialwissenschaftlichen Zeitschriften zwischen 2000 und 2009, dass die Methode der Inhaltsanalyse die dominante Methode der Kommunikationswissenschaft darstellt und in diesem Fach stärker vertreten ist als in den anderen Sozialwissenschaften (S. 36). Offen bleibt in diesem Review jedoch die Frage, wie stark automatische Verfahren in der Kommunikationswissenschaft verbreitet sind. Zwar konnte gefunden werden, dass die Erhebungsmethode nur in 2% der Fälle als computergestützte oder automatische Analyse bezeichnet wurde (S. 37), inwiefern einzelne automatische Verfahren unter anderen Bezeichnungen eingesetzt wurden, lässt sich daraus aber nicht folgern.

Die grosse Anzahl automatischer Verfahren, die im Jahr 2012 auf der Methodentagung der DGPuK zum Thema "Fortschritte in der Inhaltsanalyse" vorgestellt wurden (vgl. Sommer, Wettstein, Wirth, & Matthes, 2014: 11), weisen darauf hin, dass der Einbezug von Computern in die Inhaltsanalyse auch in der Kommunikationswissenschaft in den letzten Jahren einen Aufschwung erlebt hat. Um diese Entwicklung genauer zu untersuchen und den Umgang kommunikationswissenschaftlicher Inhaltsanalysen mit automatischen Verfahren zu beschreiben, habe ich eine eigene Inhaltsanalyse für diese Rahmenschrift durchgeführt.

Methode

Um die Frage zu beantworten, inwiefern automatische Verfahren sowohl für die Erhebung als auch im Rahmen computerunterstützter Analysen eingesetzt werden, habe ich eine standardisierte Inhaltsanalyse von Fachzeitschriftenartikeln im deutsch- und englischsprachigen Raum durchgeführt, bei der die Art der Erhebung und Auswertung im Zentrum stand. Erfasst habe ich für den deutschsprachigen Raum alle Artikel der Zeitschriften *Publizistik* und *Medien & Kommunikationswissenschaft (M&K)*, die zwischen 2004 und 2014 publiziert wurden und die Worte ***Inhaltsanalyse*** oder ***Diskursanalyse*** enthielten. Die Inhalte dieser beiden Zeitschriften wurden bereits in früheren Reviews zur Untersuchung deutschsprachiger Kommunikationswissenschaft herangezogen (vgl. Donsbach, Laub, Haas, & Brosius, 2005; Brosius & Haas, 2009). Für englischsprachliche Artikel wurden zwei top-ranked Publikationen der allgemeinen Kommunikationswissenschaft – das *Journal of Communication* und *Communication Research* – erhoben . Analog zur deutschen Stichprobe wurden hier im Zeitraum von 2004 bis 2014 alle Artikel untersucht, in denen die Begriffe ***Content Analysis*** oder ***Discourse Analysis*** auftraten.

Ergebnis

Insgesamt wurden über eine Volltextsuche 264 Artikel gefunden, von welchen 160 eigene empirische Erhebungen berichteten. Acht weitere stützten sich auf eine Sekundäranalyse inhaltsanalytischer Daten. Zwei Artikel widmeten sich methodologischen Aspekten der Inhaltsanalyse, ohne konkrete Daten zu verwenden. Die restlichen 94 Artikel enthielten lediglich eines der Suchworte oder verwiesen auf Ergebnisse von Inhaltsanalysen anderer Forscher. Für die Beschreibung der Methode der Inhaltsanalyse in der Kommunikationswissenschaft wurden nur die 160 Artikel, die eine Primärerhebung berichteten, vertieft analysiert (siehe Tabelle 1).

Tabelle 1: Anzahl untersuchter Artikel mit eigener inhaltsanalytischer Erhebung zwischen 2004 und 2014.

Zeitschrift	**Anzahl Artikel**
Publizistik	61
Medien- & Kommunikationswissenschaft	41
Journal of Communication	42
Communication Research	16
Gesamt	160

Für einen ersten Überblick über die Erhebungsmethoden in diesen Artikeln wird zunächst die Verteilung der einzelnen Verfahren über die Zeit beschrieben. In Abbildung 1 ist der Verlauf des Einsatzes quantitativer und qualitativer Inhaltsanalysen in den vergangenen zehn Jahren dargestellt. Dabei fällt einerseits auf, dass die Anzahl der inhaltsanalytischen Studien insgesamt zunimmt. Andererseits kann abgelesen werden, dass qualitative Analysen – sei es als eigenständige Erhebung oder in Kombination mit quantitativen Erhebungen – weniger stark verbreitet sind aber dennoch einen konstanten und Anteil der Studien ausmachen.

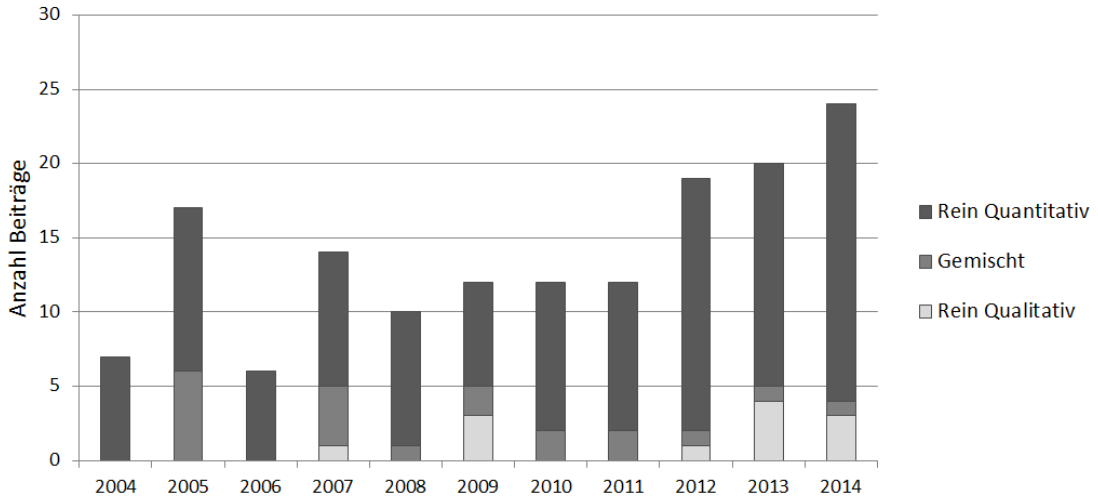


Abbildung 1: Einsatz quantitativer und qualitativer Verfahren in kommunikationswissenschaftlichen Publikationen seit 2004. Gemischte Erhebungen sind Erhebungen, die sowohl quantitative als auch qualitative Verfahren einsetzen.

Für die Erfassung des Einsatzes automatischer Verfahren wurden die Studien in vier Gruppen unterteilt. Die erste Gruppe enthält alle Studien, die sich mit der Analyse von audiovisuellem Material oder reinen Ton- und Bildquellen befassten. In diesen Studien ist eine manuelle Inhaltsanalyse heute noch alternativlos, da Verfahren zur verlässlichen automatischen Analyse dieses Materials fehlen. In die zweite Gruppe wurden jene Studien eingeteilt, die zwar Texte aus unterschiedlichen Quellen analysierten, die Analyse aber ohne den Einsatz von Computern als rein manuelle Inhaltsanalysen durchführten. Die Dritte Gruppe enthält computerunterstützte Inhaltsanalysen, in denen Computer eingesetzt wurden, um Texte für eine manuelle Analyse aus Datenbanken oder von Online-Quellen zu beschaffen. Dies war die einzige Form computerunterstützter Inhaltsanalyse in der untersuchten Stichprobe. Die vierte Gruppe sind schliesslich jene Inhaltsanalysen, welche die Texte mit Hilfe automatischer Verfahren codierten um Themen, Akteure, Frames oder andere Inhalte zu erfassen. In Abbildung 2 ist die Entwicklung der Verfahren im Zeitverlauf aufgeführt.

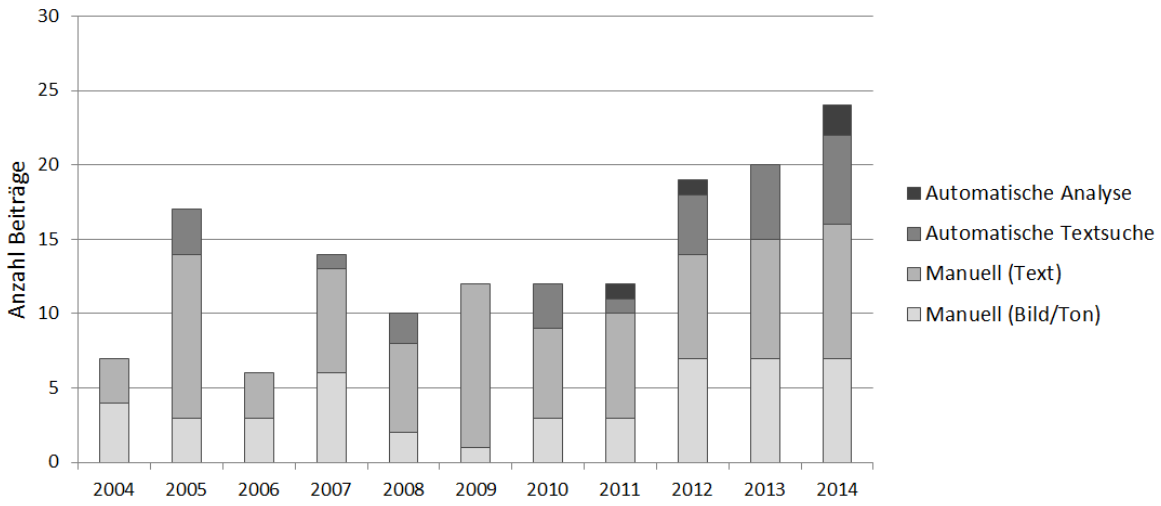


Abbildung 2: Einsatz automatischer Verfahren in kommunikationswissenschaftlichen Inhaltsanalysen seit 2004.

Tabelle 2: Kontingenztabelle der Erhebungsmethoden.

	Qualitativ	Gemischt	Quantitativ	Gesamt
Manuell (audiovisuell)	1	2	43	46
Manuell (Text)	11	13	59	83
Automatische Textsuche	0	6	20	26
Automatische Analyse	0	0	5	5
Gesamt	12	21	127	160

In Tabelle 2 ist zudem summarisch der Anteil automatischer und manueller Verfahren für qualitative und quantitative Erhebungen aufgeführt. Die Aufstellung zeigt, dass automatische Erhebungen ausschliesslich in quantitativen Inhaltsanalysen eingesetzt wurden und dass rein qualitative Analysen noch stärker als gemischte oder quantitative Inhaltsanalysen komplett auf den Einsatz von Computern verzichten.

Im Hinblick auf die Forschungsfrage, wie weit automatische Verfahren heute in der Kommunikationswissenschaft verbreitet sind, lässt sich aus diesen Ergebnissen eine klare Antwort formulieren: Sie fristen noch heute das bereits vor 14 Jahren konstatierte "Schattendasein" (Wirth & Lauf, 2001: 11). Angesichts der Tatsache, dass in unserem Fach bereits seit mehreren Jahrzehnten der Einsatz von Computern in Inhaltsanalysen diskutiert und vorgeschlagen wird (vgl. Klingemann, 1984; Bos & Tarnai, 1996), um Inhaltsanalysen zu unterstützen oder durchzuführen, ist dieses Ergebnis ernüchternd. Zwar werden vermehrt Computer eingesetzt, um Texte von Datenbanken wie LexisNexis oder von Websites und Zeitungsarchiven zu beschaffen, alle anderen Arbeitsschritte, insbesondere die Erhebung, finden jedoch weiterhin manuell statt.

Ich sehe vor allem zwei Gründe, mit denen sich die Zurückhaltung beim Einsatz automatischer Verfahren in der Kommunikationswissenschaft erklären lassen. Zum einen gibt es zwar mehrere Sammlungen automatischer Verfahren für die Textanalyse, jedoch sehr wenig Information zu automatischen Verfahren, die auch in manuellen Inhaltsanalysen gewinnbringend eingesetzt werden können. Die mangelnde Bekanntheit dieser Verfahren könnte einem Einsatz im Weg stehen. Zum anderen können Vorbehalte gegen eine vollautomatische Erhebung von Medieninhalten zu einem vollständigen Verzicht auf automatische Hilfsmittel führen. Dies, obschon eine manuelle Erhebung trotz automatischer Verfahren weiterhin möglich und in den meisten Fällen durchaus sinnvoll ist.

Das Ziel der vorliegenden Dissertation ist es deswegen, Wege aufzuzeigen, auf denen automatische Verfahren aus ihrem Schattendasein herausgeführt werden können. Dabei wird explizit nicht das Ziel verfolgt, die manuelle Inhaltsanalyse durch eine automatische zu ersetzen. Vielmehr sollen Möglichkeiten und Techniken aufgezeigt werden, durch die eine manuelle Inhaltsanalyse effizienter, verlässlicher und gründlicher durchgeführt werden kann.

2.2. Schematischer Ablauf von Inhaltsanalysen

Der Kern dieser Dissertation besteht in der Beschreibung und Anwendung von Verfahren zur Durchführung einer computerunterstützten Inhaltsanalyse. Das heisst, es werden Möglichkeiten aufgezeigt, Computer im Prozess einer Inhaltsanalyse einzusetzen, ohne damit die manuelle Codierung zu ersetzen. Um diese Verfahren systematisch in den Prozess der Inhaltsanalyse in

einzugliedern, ist es zunächst erforderlich, den Prozess einer Inhaltsanalyse und die einzelnen erforderlichen Arbeitsschritte einzuführen und zu beschreiben.

Im Wesentlichen stützt sich der hier dargestellte Prozess der Inhaltsanalyse auf die systematische Gliederung, die von Früh (2009: 102) entworfen und durch Rössler (2010: 38) auf standardisierte quantitative Inhaltsanalysen angewandt wurde. In dieser Gliederung umfasst eine Inhaltsanalyse vier Phasen (*Planungs*-, *Entwicklungs*-, *Test*- und *Anwendungsphase*, vgl. Früh, 2009: 103) mit jeweils klar definierten Arbeitsschritten. Um ein allgemeines Prozessmodell zu entwerfen, das sowohl für qualitative wie quantitative Analysen gilt, werden die Arbeitsschritte in dieser Arbeit möglichst unspezifisch formuliert und für die unterschiedlichen Herangehensweisen ausgeführt. Hierfür werden auch die allgemeinen Überlegungen zum Prozess der Inhaltsanalyse von Krippendorff (vgl. 2013: 86) und die Beschreibung qualitativer Verfahren durch Mayring (vgl. 2007) einbezogen.

Der Prozess der Inhaltsanalyse, wie er in dieser Dissertation aufgefasst wird, beginnt, nachdem ein konkretes Forschungsinteresse, eine klare Fragestellung und gegebenenfalls spezifische Hypothesen formuliert worden sind. Am Ende des Prozesses liegen inhaltsanalytische Daten vor und es folgt die Auswertung der Daten in Hinblick auf die Hypothesen und Forschungsfragen. Alle Arbeitsschritte, welche zwischen der Hypothesenbildung und Auswertung stehen, sind Teil der Inhaltsanalyse. Mit dieser Einordnung weiche ich geringfügig von früheren Prozessdefinitionen ab, in denen die Hypothesenbildung (vgl. Früh, 2009: 102; Rössler, 2010: 38), die statistische Auswertung der erhobenen Daten (vgl. Früh, 2009; Krippendorff, 2013: 86) oder die Darstellung und Interpretation der Ergebnisse (vgl. Mayring, 2007: 14f; Krippendorff, 2013: 86) ebenfalls als Teil des Prozesses einer Inhaltsanalyse dargestellt wurden.

2.2.1. Planungsphase

Die erste Phase des Forschungsprozesses ist die Planungsphase oder Konzeption der Inhaltsanalyse (vgl. Früh, 2009: 103). Hier findet die Operationalisierung der zentralen Konzepte der Forschungsfrage und die Planung der Erhebung statt, mit der Daten für ihre Beantwortung gesammelt werden sollen. Grundvoraussetzung für diese Phase ist also ein klares Forschungsinteresse, welches die Art der Information, der Texte und des erkenntnistheoretischen Ansatzes für die Inhaltsanalyse vorgibt. Zu diesem konkreten Forschungsinteresse zählen gerade bei quantitativen Fragestellungen auch ausformulierte Hypothesen. Die Planungsphase lässt sich in drei, teilweise voneinander abhängige, Arbeitsschritte gliedern, die alle noch vor dem Beginn der Entwicklungsphase abgeschlossen sein müssen.

Texte

Ein weiterer wichtiger Schritt in dieser Phase ist die Planung des zu verwendenden Textkorpus aus der Grundgesamtheit aller denkbaren Texte. Jung (2001: 33f) nennt in diesem Zusammenhang mit den Kommunikationsbereichen, Teildiskursen und Textsorten, drei Dimensionen, entlang derer der Forscher Entscheidungen treffen muss. Zudem muss auch entlang der zeitlichen Dimension eine Einschränkung gemacht werden, um nur Texte aus einem bestimmten Zeitraum in die Analyse einzubeziehen. Falls die Anzahl der relevanten Texte nach diesen Einschränkungen zu hoch ist, kann in diesem Schritt zudem eine Stichprobe der Quellen, Zeiträume oder Texte gezogen werden.

In der Dimension der *Kommunikationsbereiche* muss die Art der Kommunikation definiert werden, auf die sich die Forschungsfrage bezieht. Beispiele für solche Kommunikationsbereiche sind Fachkommunikation, interne Organisationskommunikation, Massenkommunikation, Werbung oder Online-Diskussionen.

In der Dimension der *Teildiskurse* muss die Kommunikation auf bestimmte Inhalte eingegrenzt werden, welche für das Forschungsinteresse zentral sind. Diese Teildiskurse können entweder spezifische Unterthemen eines Themenbereiches (vgl. Jung, 2001: 34) oder auch Texte im Zusammenhang mit einem bestimmten Ereignis (vgl. Keller, 2001: 137f) sein.

In der Dimension der *Textsorten* muss eingegrenzt werden, welche Art der Texte aus einem bestimmten Kommunikationsbereich zu einem bestimmten Teildiskurs untersucht werden sollen (vgl. Rössler, 2010: 54). Im Fall von Massenkommunikation kann hier zwischen unterschiedlichen Trägermedien, Beitragstypen oder Medienangeboten unterschieden werden (z.B: Artikel und Kommentare in den reichweitenstärksten überregionalen Zeitungen: Gerhards, 2003: 304), in der Organisationskommunikation zwischen Übertragungsarten (z.B: mündlich, schriftlich, elektronisch, top-down vs. bottom-up).

In der Dimension der *Zeit* kann schliesslich eingegrenzt werden, in welchen Zeiträumen die Texte für die Analyse erscheinen müssen, um für die Analyse relevant zu sein. Hier kann entweder ein einzelner, klar definierter Zeitraum bestimmt werden (z.B: Ein Jahr ab dem Reaktorunglück in Fukushima) oder eine Reihe von kurzen Zeiträumen im Umfeld von Ereignissen (z.B: Jeweils vier Wochen vor jeder Kommunalwahl).

Neben dieser Eingrenzung des relevanten Textkorpus in mehreren Dimensionen muss in diesem Schritt der Planung auch überlegt werden, wie einzelne relevante Texte in ihrer natürlichen Einheit gefunden werden können und ob gegebenenfalls nur einzelne Teile analysiert werden müssen. So kann beispielsweise definiert werden, dass jeweils nur die Titelseiten der Zeitungen, nur die ersten

10 Minuten der Diskussionen oder nur die Anmoderationen der Beiträge inhaltsanalytisch erfasst werden.

Erhebungsmethode

In der Planung der Erhebungsmethode müssen alle Konzepte, die in der Forschungsfrage und den Hypothesen genannt sind, so operationalisiert werden, dass sie über eine Inhaltsanalyse aus Texten extrahiert werden können. Früh (2009: 118) spricht in diesem Zusammenhang von *Indikatoren* in Texten, die mit unterschiedlicher Deutlichkeit auf einen bestimmten Inhalt hinweisen können. Dies können einzelne Schlüsselbegriffe oder auch generelle Ideen oder Schemata sein, die im Text manifest werden. Ein einzelnes Konzept muss nicht zwingend durch einen einzigen Indikator repräsentiert sein, sondern kann auch durch die Kombination mehrerer Indikatoren operationalisiert werden (vgl. Krippendorff, 2013: 194; Semetko & Valkenburg, 2000).

Ausgehend vom Forschungsinteresse und der Indikatoren lässt sich bestimmen, welche Art der Erhebung für die vorliegende Analyse in Frage kommt. In diesem Schritt kann eine Entscheidung zwischen qualitativer und quantitativer Erhebung, sowie eine Entscheidung für oder gegen computergestützte Verfahren in der Analyse getroffen werden.

Dabei ist zu bedenken, dass diese Optionen sich nicht zwingend gegenseitig ausschliessen, da quantitative Analysen auch qualitative Bestandteile enthalten können (vgl. Gerhards, 2003: 306), und qualitative Erhebungen auch mit quantitativen Methoden ergänzt werden können (vgl. Mayring 2007: 45). Genauso können manuelle und automatische Verfahren in unterschiedlichen Formen kombiniert werden (siehe Kapitel 3).

Administration

Der dritte Arbeitsschritt in der Planungsphase einer Inhaltsanalyse ist die Planung der Administration, der Finanzierung und der Arbeitsteilung für die gesamte Inhaltsanalyse (vgl. Rössler, 2010: 47). Dieser Arbeitsschritt setzt voraus, dass Art und Umfang des Textkorpus abgeschätzt werden können und die Erhebungsmethode definiert ist.

Zunächst gilt es in diesem Schritt, den Umfang der Inhaltsanalyse abzuschätzen und gegebenenfalls den finanziellen und zeitlichen Rahmenbedingungen des Projektes anzupassen (vgl. Rössler, 2010: 47). Ausgehend von der Art und dem Umfang der zu analysierenden Texte kann berechnet werden, wie viele Texteinheiten (z.B: Artikel, Seiten, Sendeminuten, Diskussionsprotokolle oder Bilder) gesamthaft vorhanden sind. Gleichzeitig lässt sich anhand der Definition und Anzahl der Indikatoren abschätzen, wie lange die Bearbeitung einer einzelnen Texteinheit dauern wird. Gegebenenfalls kann hier eine Voranalyse durchgeführt werden, um verlässliche Schätzungen zu erhalten.

Aufbauend auf dieser ersten Schätzung kann anschliessend errechnet werden, wie viele Arbeitsstunden insgesamt auf alle Mitarbeiter entfallen und ob die Gesamtheit der relevanten Texte oder nur eine Stichprobe analysiert werden kann. Gleichzeitig ergibt sich aus dieser Schätzung ein Anhaltspunkt für den Bedarf an Codierern und Mitarbeitern für die Inhaltsanalyse.

Zudem muss in diesem Arbeitsschritt die technische Machbarkeit der Inhaltsanalyse mit den gesetzten Rahmenbedingungen geprüft werden. Dazu gehört die Abklärung der Verfügbarkeit der Texte, das Vorgehen bei der Textakquise und allenfalls benötigte technische Hilfsmittel für die Beschaffung und Analyse des Untersuchungsmaterials.

Schliesslich sollte auch ein Zeitplan für den gesamten Prozess der Inhaltsanalyse erstellt werden. Dabei muss bedacht werden, dass einzelne Arbeitsschritte parallel bearbeitet werden können, während andere den Abschluss vorangegangener Arbeitsschritte bedingen.

Das Ergebnis der Planungsphase mit ihren drei interdependenten Arbeitsschritten besteht aus drei unabhängigen und konkreten Ausarbeitungen. Erstens liegt eine Suchstrategie für Texte vor, die neben Suchbegriffen und Details zum Zugang und Zugriff auf Archive auch die Kriterien enthält, was einen relevanten Text ausmacht. Zweitens liegt eine erste Operationalisierung aller benötigten Indikatoren vor, die in der Entwicklungsphase feiner ausgearbeitet werden kann. Drittens steht ein Zeitplan für die verbleibenden Arbeitsschritte und eine Abschätzung des Arbeitsaufwands, Personalbedarfs und der Kosten. Mit diesen Informationen kann die Arbeit an der Entwicklungsphase aufgenommen werden.

2.2.2. Entwicklungsphase

Die zweite Phase im Prozess einer Inhaltsanalyse ist die Entwicklungs- oder Definitionsphase, in der das Erhebungsinstrument, das Textkorpus und das Personal zusammengestellt werden (vgl. Früh 2007). Sie lässt sich in vier Arbeitsschritte einteilen, wobei vor allem die Entwicklung des Erhebungsinstruments und die Probecodierung stark voneinander abhängen und gleichzeitig ausgeführt werden müssen.

Erhebungsinstrument

Das Erhebungsinstrument ist eine Sammlung von Regeln, die in der Anwendungsphase auf die zu analysierenden Texte angewandt werden, um relevante Informationen zu extrahieren. Je nach Art der Inhaltsanalyse kann dieses Instrument unterschiedliche Formen annehmen. In der quantitativen Inhaltsanalyse ist es ein Codebuch mit exakten Codiervorschriften für die Analyse der einzelnen Codiereinheiten. Bei qualitativen Analysen ist das Erhebungsinstrument ein Leitfaden für die Analyse

der Texte und bei automatischen Analysen wird das Erhebungsinstrument nicht nur in natürlicher Sprache ausformuliert, sondern muss als Programm implementiert werden.

Das Codebuch quantitativer Inhaltsanalysen enthält ein ausformuliertes Kategoriensystem (vgl. Rössler, 2010: 97; Gerhards, 2003: 306f), das vorgibt, welche Eigenschaften der Texte codiert werden sollen und nach welchen Regeln Texteinhalte in numerische Werte umgewandelt werden. Es gibt also vor, welche Elemente gezählt werden sollen und in welche Kategorien semantische Elemente klassifiziert werden müssen. Für die Definition der Kategorien ist teilweise eine qualitative Voranalyse nötig, in der das Kategoriensystem aufgrund einer repräsentativen Auswahl von Texten erarbeitet wird (vgl. Wessler, 1999).

Der Leitfaden einer qualitativen Analyse regelt das genaue Vorgehen bei der Lektüre von Texten und der Definition von Ober- und Unterkategorien. Zudem werden dort Regeln für die Notation und Aggregation der Ergebnisse aufgestellt (vgl. Mayring, 2007: 53ff), die für eine nachvollziehbare Codierung eingehalten werden müssen. Auch die Art der Querverweise oder direkten Zitation von Textstellen wird im Leitfaden vorgegeben.

Die Umsetzung der Regeln für automatische Erhebungen kann entweder durch die Programmierung neuer Algorithmen oder durch Anpassung bereits bestehender Programme erfolgen. In beiden Fällen müssen die aufgestellten Regeln zur Identifikation und Analyse der Indikatoren so präzise wie möglich implementiert werden, um systematische Fehler in der Erhebung zu vermeiden. Je nach Art der Indikatoren bieten sich andere automatische Verfahren für die Erhebung an (vgl. Alexa, 1997; Popping, 2000).

Textakquise

Ein weiterer aufwändiger Arbeitsschritt in der Entwicklungsphase ist der Aufbau des Textkorpus für die Inhaltsanalyse, also die Bereitstellung des Untersuchungsmaterials. Diese Arbeit hat sowohl eine konzeptionelle als auch eine physische Komponente.

Konzeptionell muss bestimmt werden, welches die Analyseeinheit für die vorliegende Inhaltsanalyse ist. Dies kann entweder der komplette Text, Segmente des Textes (zum Beispiel einzelne Abschnitte, einzelne Kurznachrichten oder Sendeminuten), oder semantische Einheiten innerhalb des Textes (z.B: Argumente, Problemdefinitionen, Angriffe oder Symbole) sein. In einzelnen Fällen können auch verschachtelte oder hierarchische Analyseeinheiten definiert werden, so dass sowohl der Text als auch einzelne Segmente und semantische Einheiten gleichzeitig analysiert werden (vgl. Rössler, 2010: 78). Eine solche hierarchische Gliederung von Analyseeinheiten stellt besondere Anforderungen an die Entwicklung des Erhebungsinstruments und macht Überlegungen

zur maximalen Anzahl von Analyseeinheiten nötig, die auf unterschiedlichen Ebenen innerhalb eines Textes oder eines Textabschnitts untersucht werden sollen (vgl. Wettstein, Wirth, & Reichel, 2015).

Physisch muss das Textkorpus beschafft und für die Analyse bereitgelegt werden. Die Art der Ablage unterscheidet sich je nach untersuchter Textsorte und kann digital oder als Sammlung von Zeitungen, Kassetten oder Gesprächsnotizen vorliegen. Wichtig ist, dass die Ablage so gestaltet wird, dass die Texte nach diesem Schritt direkt für die Analyse verwendet werden können und keiner weiteren Bearbeitung mehr bedürfen. Falls dies erforderlich oder sinnvoll ist, können die Texte bereits auf die richtige Analyseeinheit reduziert und mit einer eindeutigen Identifikation versehen werden, damit am Ende der Analyse jede Codierung eindeutig einem Textabschnitt zugewiesen werden kann.

Probecodierung

Ein weiterer wichtiger Teil der Entwicklungsphase ist der Test des Erhebungsinstruments durch den Forscher selber. In diesen Tests wird das Erhebungsinstrument erstmals auf die *Anwendbarkeit* auf die Texte, die *Eindeutigkeit* und *Vollständigkeit* der Anweisungen und die *Validität* der Messung geprüft (vgl. Rössler, 2010: 101). Zu diesem Zweck werden einzelne Texte aus dem vorbereiteten Textkorpus analysiert.

In Bezug auf die Anwendbarkeit muss geprüft werden, ob das Erhebungsinstrument sich auf die Texte anwenden lässt und eine Analyse überhaupt möglich ist. Gerade bei automatischen Verfahren ist in diesem Zusammenhang zu prüfen, ob das Dateiformat der Texte stimmt und die Programme sie korrekt einlesen.

In Bezug auf die Eindeutigkeit muss geprüft werden, ob das Instrument für jeden der analysierten Texte genau vorgibt, welche Inhalte extrahiert werden müssen, wie die Kategorisierung vorgenommen werden kann und welche Codes vergeben werden sollen. Treten hier ambige Situationen auf, bei welchen ein Text unterschiedliche Entscheidungen rechtfertigen würde, muss das Erhebungsinstrument präzisiert werden (vgl. Früh, 2009: 120).

Die Vollständigkeit sollte in zweifacher Hinsicht überprüft werden. Zum einen muss festgestellt werden, ob alle in den Texten vorkommenden relevanten Informationen tatsächlich durch das Erhebungsinstrument erfasst werden können. Dies ist insbesondere da wichtig, wo mit Listen von Themen, Akteuren, Stilelementen, oder Aussagen gearbeitet wird (vgl. Rössler 2010: 101). Zum anderen muss in Hinblick auf das Forschungsinteresse überprüft werden, ob alle Informationen, welche zur Beantwortung der Forschungsfrage nötig sind, extrahiert werden (vgl. Krippendorff, 2013: 184f).

Schliesslich muss auch die Validität geprüft werden. Zu diesem Zweck können die Ergebnisse der Analyse kritisch mit anderen Quellen verglichen werden, um sicherzustellen, dass die erhaltenen Daten tatsächlich die gesuchten Konstrukte abbilden. Hajer (2003: 282) schlägt hierfür vor, die Ergebnisse anhand von Interviews mit Kommunikatoren und Experten zu validieren. Es können jedoch auch vergleichbare Analysen anderer Forscher (vgl. Mayring, 2007: 111) zur Validierung der Messung herangezogen werden. Widerspricht die Abstraktion der Texte den Aussagen vergleichbarer Analysen desselben Materials, so müssen gegebenenfalls die Definitionen des Erhebungsinstruments oder die Konzeption der Indikatoren angepasst werden.

Falls die Probecodierung Änderungen am Erhebungsinstrument nötig macht, muss dieses anschliessend erneut getestet werden. Gerade im Fall automatischer Analysen müssen die Programme meist mehrfach geprüft und überarbeitet werden, bis sie valide Ergebnisse liefern. Dieser Prozess kann mitunter Monate dauern. Auch für manuelle quantitative Inhaltsanalysen können jedoch mehrere Testrunden nötig sein, bis ein valides Erhebungsinstrument entwickelt ist (vgl. Gerhards, 2003: 317).

Rekrutierung

Die administrative Arbeit in der Entwicklungsphase besteht primär in der Anwerbung und Anstellung von Codierern und gegebenenfalls weiteren benötigten Mitarbeitern für die Anwendungsphase. Je nach Anzahl der eingeplanten Codierer kann dieser Arbeitsschritt Ausschreibungen, Bewerbungsgespräche und Eignungstests beinhalten, für welche ebenfalls Zeit eingeplant werden muss.

Die Entwicklungsphase ist abgeschlossen, wenn ein vollständiges Textkorpus, ein funktionierendes Erhebungsinstrument und ausreichend Codierer und Mitarbeiter für die Erhebung zur Verfügung stehen. Mit diesen Zwischenergebnissen kann die Testphase der Inhaltsanalyse eingeleitet werden.

2.2.3. Testphase

Die Testphase ist eng mit der Entwicklungsphase der Inhaltsanalyse verbunden und kann erneut zur Überarbeitung des Erhebungsinstruments oder des Textkorpus führen. In diesem Schritt wird die Inhaltsanalyse unter realen Bedingungen getestet. Es werden also erstmals Codierer mit den Texten und dem Instrument konfrontiert, die an der Entwicklung der Inhaltsanalyse nicht beteiligt waren. Die Testphase umfasst mit der Schulung dieser Codierer und einem Test ihrer Eignung zwei Arbeitsschritte.

Codiererschulung

Die Codiererschulung kann je nach Umfang und Art der Inhaltsanalyse entweder eine tatsächliche, mehrtägige Schulung von Hilfskräften sein, um ihnen das Erhebungsinstrument im Detail zu erläutern, oder eine kurze Einweisung in einen Leitfaden oder ein Computerprogramm, welches als Erhebungsinstrument dient. In Extremfällen kann eine Codiererschulung auch mehrere Monate dauern (vgl. Gerhards, 2003: 317), falls das Erhebungsinstrument sehr komplex ist oder Definitionen wiederholt angepasst werden müssen.

Ziel der Codiererschulung ist es, Mitarbeiter für den Umgang mit dem Erhebungsinstrument und den vorbereiteten Texten auszubilden, damit sie in der Anwendungsphase selbständig arbeiten und Daten erheben können.

Eignungstest

Die Testphase wird in der Regel durch einen Reliabilitäts- und Validitätstest der Codierentscheidungen abgeschlossen. Beide Tests bestehen aus der Codierung einer gewissen Anzahl von Texten durch die geschulten Codierer. Die Codierungen werden anschliessend verglichen und auf Übereinstimmung untereinander und mit einer durch den Forscher erstellten Musterlösung verglichen (vgl. Mayring, 2007: 112f; Krippendorff, 2013: 273). Im Fall von quantitativen Analysen kann hier die exakte prozentuale Übereinstimmung berechnet werden, bei qualitativen Analysen muss ein inhaltlicher Vergleich der resultierenden Interpretationen vorgenommen werden.

Weichen einzelne Codierer in mehreren Entscheidungen von der Musterlösung und den anderen Codierern ab, so weist dies auf eine mangelnde Validität der Codierentscheidungen dieser Codierer hin und kann eine Nachschulung erforderlich machen (vgl. Früh, 2009: 103). Weichen mehrere Codierer bei einzelnen Entscheidungen oder Arbeitsschritten stark voneinander ab, so weist dies auf ein zu wenig präzises Erhebungsinstrument hin und macht eine Überarbeitung des Instruments für die betreffenden Kategorien nötig. Stimmen alle Codierer in ihren Entscheidungen hinreichend mit der Musterlösung und untereinander überein, so kann das Instrument als reliabel angenommen werden.

Bei automatischen Analysen kann eine Übereinstimmung von 100% erwartet werden, welche erreicht wird, falls alle Codierer die Software korrekt einsetzen und es sich nicht um trainierte Verfahren handelt, deren Entscheidungen sich neuen Informationen anpassen können. Für quantitative Inhaltsanalysen gibt es vage definierte Richtwerte, die zwar weitgehend akzeptiert aber nicht universell gültig sind (vgl.: Lombard et al., 2002: 593). Bei qualitativen Inhaltsanalysen liegt es im Ermessen des Forschers, wie gross der Interpretationsspielraum bei der Analyse der Texte liegen darf, um die Validität der Erhebung zu gewährleisten (vgl. Flick, 1987; Schwab-Trapp, 2003: 183).

Das Ergebnis der Testphase ist ein Team gut ausgebildeter Codierer, die das Erhebungsinstrument auf die Texte anwenden können, verlässliche Codierentscheidungen treffen und valide Daten produzieren. Unter Umständen können Erfahrungen in der Testphase einen Rückgriff auf die Entwicklungsphase nötig machen, wenn einzelne Teile des Erhebungsinstruments präzisiert oder abgeändert werden müssen, um eine verlässliche Codierung sicherzustellen.

2.2.4. Anwendungsphase

Mit dem Abschluss der Entwicklungs- und Testphase kann mit der Anwendungsphase die finale Phase der Inhaltsanalyse beginnen. In dieser Phase wird das Erhebungsinstrument auf die Texte angewandt, um die gewünschte Information zu gewinnen und in einer geeigneten Form zu speichern (vgl. Früh, 2009: 103). In dieser Phase wird der grösste Teil der Arbeit durch die Codierer übernommen, dennoch bleiben einzelne administrative und abschliessende Arbeitsschritte für den Forscher.

Erhebung

Die Erhebung findet statt, indem trainierte Codierer das Erhebungsinstrument auf das vorbereitete Textkorpus anwenden. Dieser Arbeitsschritt läuft weitgehend autonom ab, falls das Training der Codierer erfolgreich und die Definition des Erhebungsinstruments eindeutig ist.

Der Zeitraum, in welchem dieser Arbeitsschritt stattfindet, kann je nach Umfang der Analyse von wenigen Tagen bis zu mehreren Jahren umfassen. Bei qualitativen Analysen ist dies mitunter der langwierigste Arbeitsschritt (vgl. Schwab-Trapp, 2003: 174). Jedoch kann er auch bei quantitativen Inhaltsanalysen mit grossen Textkorpora viel Aufwand bedeuten. Bei automatischen Analysen ist die eigentliche Erhebung dagegen der mit Abstand kürzeste Arbeitsschritt und kann je nach Analyse in wenigen Minuten oder Stunden abgeschlossen sein.

Die Erfassung und Speicherung der extrahierten Information kann entweder auf einem vorbereiteten Codesheet aus Papier oder in einer elektronischen Tabelle am Computer stattfinden. Zur Eingabe kann sowohl bei quantitativen als auch bei qualitativen Inhaltsanalysen auch auf geeignete Software zurückgegriffen werden, welche die Dateneingabe vereinfacht (vgl. Luzar, 2004: 148; Macnamara, 2005).

Betreuung

Auch wenn die Erhebung zu grossen Teilen selbständig durch die Codierer durchgeführt wird, müssen sie in mehrerer Hinsicht betreut werden. Insbesondere muss in diesem Arbeitsschritt Hilfestellung bei Problemen in Zusammenhang mit der Erhebung geleistet, die Texte an die

jeweiligen Codierer verteilt und die geleistete Arbeitszeit erfasst werden. Dieser Arbeitsschritt kann bei grossen Codierteams einen erheblichen Aufwand bedeuten, der in der Zeitplanung bedacht werden muss.

Kontinuierlicher Reliabilitäts- und Validitätstest

Gerade in Inhaltsanalysen mit langer Laufzeit sind regelmässige, kurze Tests der Reliabilität und Validität der Codierung wichtig. Damit kann sichergestellt werden, dass auch im Verlauf der Inhaltsanalyse die Qualität der Daten erhalten bleibt. Für diese Tests kann entweder analog zur Testphase verfahren werden oder es kann ein verdeckter Reliabilitätstest durchgeführt werden, bei dem die Codierer sich der Testsituation nicht bewusst sind und Test-Texte im Rahmen ihrer täglichen Arbeit analysieren (vgl. Rössler, 2010: 197).

Gleichzeitig können auch während der Analyse Tests auf Validität der Ergebnisse vorgenommen werden, indem sie mit Daten aus anderen Quellen verglichen oder Experten zur Diskussion und Prüfung vorgelegt werden (vgl. Hajer, 2003: 283).

Datenaggregation und -Aufbereitung

Ein weiterer Arbeitsschritt in der Anwendungsphase ist die regelmässige Sammlung, Speicherung und Aufbereitung der anfallenden Daten. Falls die Codierer die Daten auf Papier erheben, zählt dazu auch die elektronische Erfassung der Codierbögen für die spätere Auswertung. Im Fall von digital erhobenen Daten ist auf jeden Fall eine regelmässige Sicherung der Daten angeraten, um keine Arbeit zu verlieren.

Zur Aufbereitung gehört je nach Art der erhobenen Daten auch eine Eingabe in geeignete Statistikprogramme und die Beschriftung und Definition der Variablen des Datenblatts. Gerade bei quantitativen Inhaltsanalysen, in denen eine Vielzahl von Kategorien erhoben wird, ist die Datenstruktur frühzeitig zu bedenken und vorzubereiten.

Das Ergebnis der Anwendungsphase ist zugleich das Ergebnis der gesamten Inhaltsanalyse. Es besteht einerseits aus Daten, welche eine Abstraktion der untersuchten Texte darstellen und für die Beantwortung der Forschungsfrage herangezogen werden können, andererseits aus einer Dokumentation des Prozesses und der Qualität der Erhebung (vgl. Krippendorff, 2013: 86).

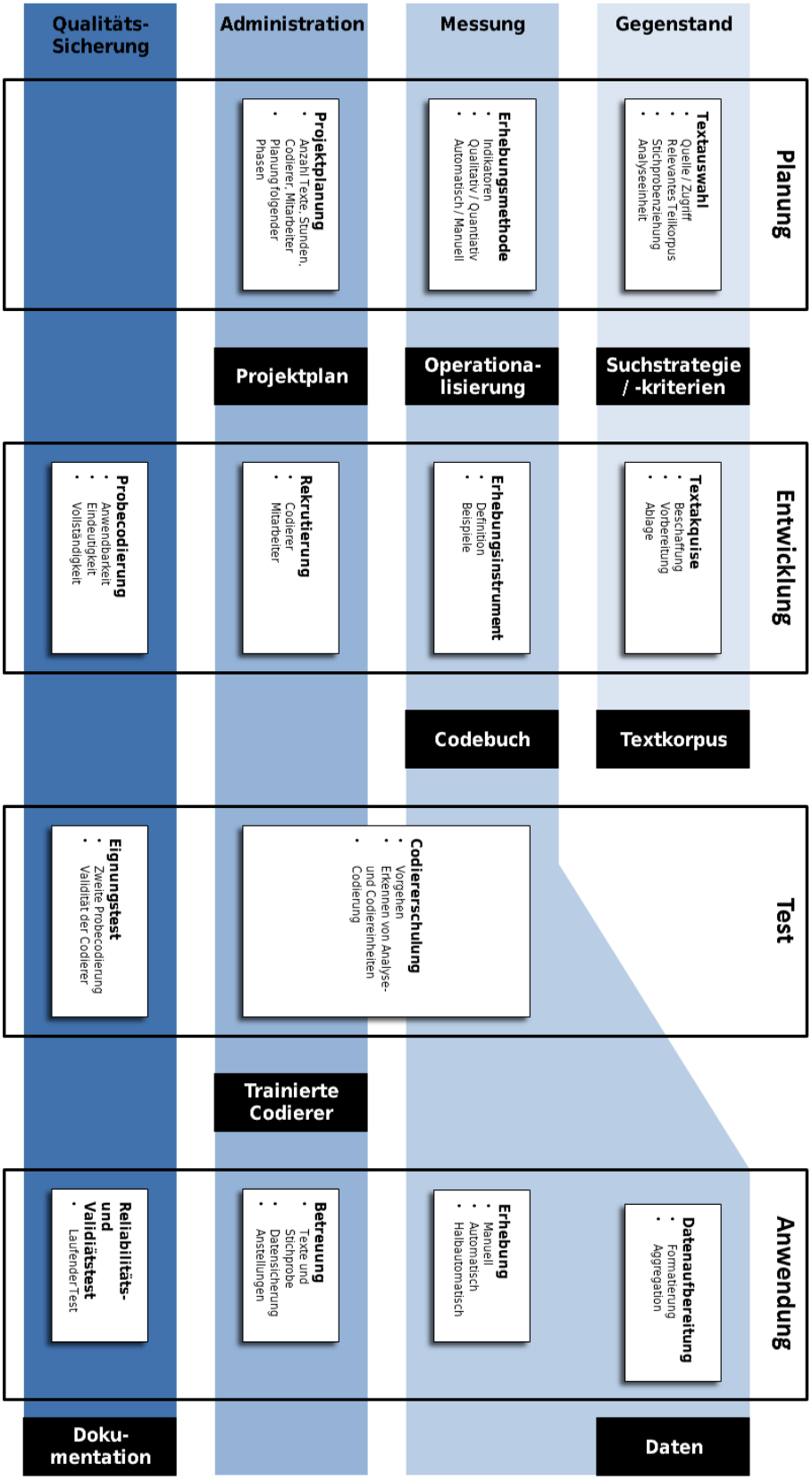


Abbildung 3: Graphische Darstellung des allgemeinen Prozesses einer Inhaltsanalyse. Der Prozess gliedert sich in vier Aufgabenbereiche (Gegenstand, Messung, Administration und Qualitätssicherung) und in vier voneinander getrennte Phasen mit jeweils eigenen Ergebnissen (schwarze Kästen). Die Arbeitsschritte, welche für jeden Aufgabenbereich in den jeweiligen Phasen anfallen, sind als weisse Kästen dargestellt.

Kapitel 3. Computerunterstützte Inhaltsanalyse

Die computerunterstützte Inhaltsanalyse, wie sie in dieser Dissertation vorgestellt wird, ist im Grunde eine manuelle Inhaltsanalyse, da die finalen Codierentscheidungen durch menschliche Codierer getroffen werden. Computer werden jedoch im Prozess der Inhaltsanalyse an unterschiedlichen Stellen eingesetzt, um sowohl den Forscher als auch den Codierer zu unterstützen und die Effizienz und Qualität der Datenerhebung zu verbessern.

Mit dieser Definition schliesse ich an frühere Unterscheidungen zwischen computergestützter oder automatischer Inhaltsanalyse und computerunterstützter Inhaltsanalyse an (vgl. Stuckardt, 2000: 25ff; Scharkow, 2012: 45f), welche die Trennung zwischen diesen Verfahren an der finalen Codierentscheidung festmacht. Diese für die vorliegende Arbeit sinnvolle Unterscheidung grenzt sich deutlich von früheren Definitionen ab, die jeden Einbezug automatischer Verfahren in eine Inhaltsanalyse als computerunterstützte Analyse bezeichneten, unabhängig davon ob die Codierentscheidungen schliesslich von Menschen oder Computern gefällt wurden (vgl. Klingemann, 1984; Bos & Tarnai, 1996; Alexa, 1997; Popping, 2000).

In der manuellen Inhaltsanalyse werden unterschiedliche Ansätze diskutiert und angewandt, bei denen automatische Verfahren in die Inhaltsanalyse einbezogen werden. In dieser Arbeit unterscheide ich bei diesen Ansätzen zwischen substituierenden, parallelen und halbautomatischen Ansätzen. Beim *parallelen* Einsatz automatischer Verfahren wird sowohl ein Erhebungsinstrument für die manuelle Erhebung als auch ein Instrument für die automatische Erhebung entwickelt. In der Anwendungsphase finden damit zwei voneinander unabhängige Erhebungen statt, die jeweils eigene Daten produzieren. Die unterschiedlichen Datensätze werden im Zuge der Datenaufbereitung einer einzigen Datenbank verbunden. Dieser Ansatz kann zur Anwendung kommen, wenn einzelne Indikatoren wie die Länge oder das Erscheinungsdatum von Texten schnell und einfach automatisch erfasst werden können, gleichzeitig aber inhaltliche Kategorien erhoben werden müssen, für die nicht auf das Weltwissen eines menschlichen Codierers verzichtet werden kann.

Beim *substituierenden* Ansatz der computerunterstützten Inhaltsanalyse werden einzelne Arbeitsschritte oder ihre Bestandteile im Prozess der Inhaltsanalyse durch automatische Verfahren ersetzt. Die am weitesten verbreitete Anwendung dieses Ansatzes besteht in der automatischen Suche nach relevanten Texten über bestimmte Schlüsselworte. Zusätzlich kann auch die Formatierung, Beschriftung und Verwaltung von Texten im Korpus und deren Distribution an Codierer automatisiert werden. Auch automatische Auswertungen der Reliabilität und Validität von Codierungen sind meist mit wenig Aufwand realisierbar.

Beim *halbautomatischen* Ansatz, der im Rahmen dieser Dissertation formuliert und in die Praxis umgesetzt wurde, wird die manuelle Erhebung durch automatische Verfahren unterstützt, um die manuelle Codierung insgesamt effizienter zu gestalten (vgl. Wettstein, 2014b). Bei diesem Ansatz findet die manuelle Erhebung über eine geeignete Eingabemaske statt, die das Lesen der Texte und die Eingabe der Daten am Bildschirm erlaubt. Im Hintergrund kann diese Eingabemaske mit automatischen Verfahren verknüpft werden, um den Codierer bei seiner Arbeit durch Hinweise und Hilfestellungen zu unterstützen (siehe Absatz 7.1.2).

Die meisten Computerprogramme für die Durchführung computerunterstützter Inhaltsanalysen sind Verfahren zur automatischen Textanalyse, die aus Anwendungen der automatischen Inhaltsanalyse bekannt sind. Sie lassen sich grob in drei Kategorien unterteilen, auf die ich mich in nachfolgenden Kapiteln beziehen werde. Eine erste Kategorie bilden *explorative textstatistische Verfahren*, bei denen das Auftreten von Begriffen in Kontingenzanalysen ausgewertet wird. Oft werden grafische Darstellungen oder multidimensionale Skalierungsmethoden verwendet, um sich ein Bild über den Inhalt von Texten zu verschaffen (vgl. Alexa, 1997: 21-24; Popping, 2000: 24). Diese Methode kann einerseits in automatischen Inhaltsanalysen zur explorativen Beschreibung eines Diskurses (vgl. Miller & Riechert, 2001; Crawley, 2007) oder in computerunterstützten Inhaltsanalysen zur Ergänzung zu einer qualitativen Vorstudie (vgl. Wettstein, 2012b) durchgeführt werden.

Eine zweite Kategorie bilden *trainierte textstatistische Verfahren*, die eingesetzt werden können, um unbekannte Texte anhand eines Trainingskorpus manuell codierter Texte automatisch zu klassifizieren. Dabei werden die Begriffe, welche in den unbekannten Texten vorkommen, mit der Verteilung von Begriffen in unterschiedlich klassifizierten Trainingstexten verglichen, um die Wahrscheinlichkeit ihrer Zugehörigkeit zu unterschiedlichen Klassen zu bestimmen. Während diese Verfahren auch eingesetzt werden können, um einzelne Kategorien in einer Inhaltsanalyse automatisch zu bestimmen (vgl. Scharkow, 2012: 202), ist ihre Verwendung in der Vorbereitung von Inhaltsanalysen, namentlich in der Abschätzung der Relevanz einzelner Texte für die Inhaltsanalyse, für eine computerunterstützte Inhaltsanalyse besonders interessant.

Eine dritte – äusserst heterogene – Kategorie wird durch die *deduktiven* oder *regel- und diktionärsbasierten Verfahren* gebildet. Bei diesen Verfahren werden durch den Forscher Regeln und Schlüsselbegriffe definiert, anhand derer die Kategorien einer Inhaltsanalyse automatisch codiert werden können. Im einfachsten Fall, der Volltextsuche von Textkorpora, können diese Regeln nur in der dichotomen Entscheidung bestehen, ob ein Schlüsselwort in einem Text auftritt. In etwas fortschrittlicheren Verfahren dieser Kategorie können einzelne Bestandteile eines Textes

automatisch ausgelesen werden, um beispielsweise das Datum, die Länge, den Titel oder andere Kategorien automatisch zu erheben. Schliesslich können diese Verfahren auch verwendet werden, um komplexe grammatikalische und semantische Zusammenhänge zwischen Begriffen zu erkennen und damit die Satzstruktur und die Bedeutung von Aussagen aufzuschlüsseln oder zu paraphrasieren (vgl. Roberts, 1989; van Atteveldt, Kleinnijenhuis, & Ruigrok, 2008; Wüest, Clematide, Bünzli, Laupper, & Frey, 2011).

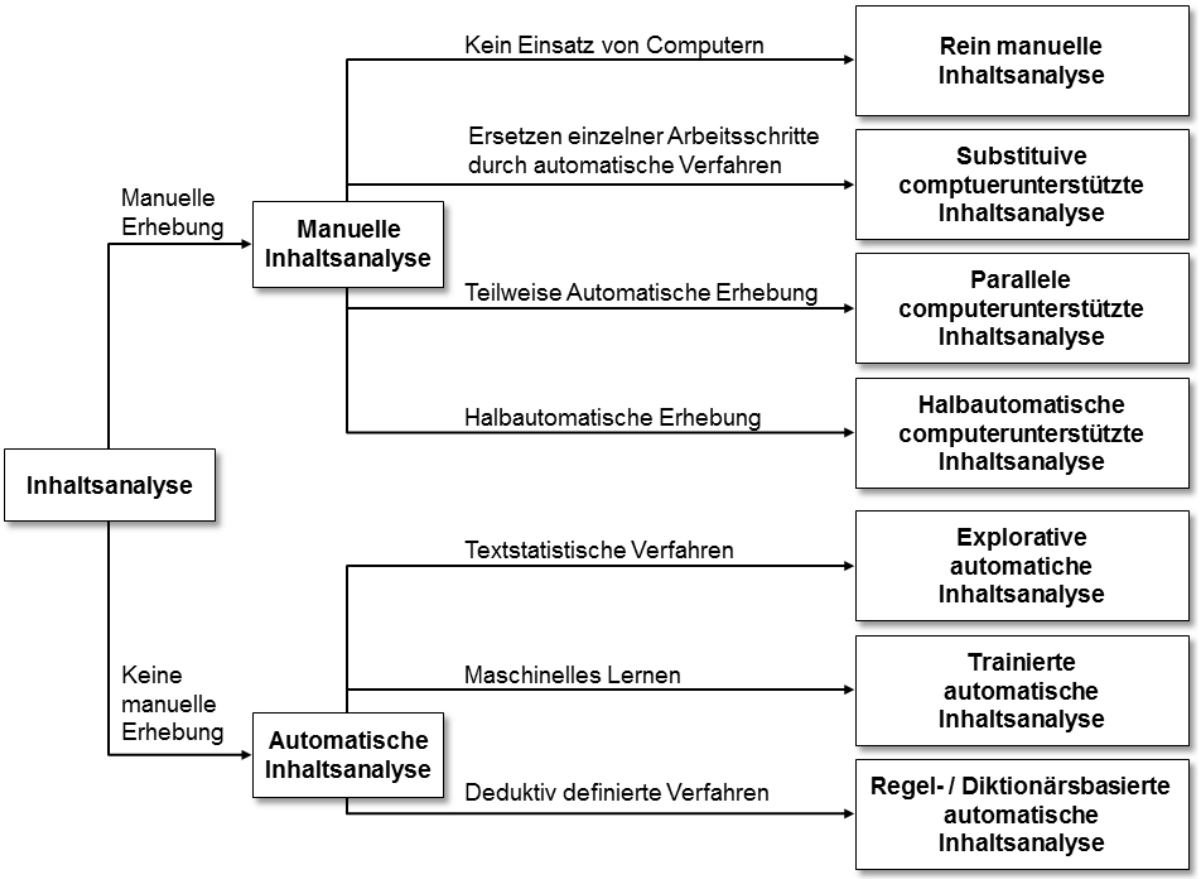


Abbildung 4: Überblick der Kategorisierung unterschiedlicher inhaltsanalytischer Ansätze mit und ohne Einbindung automatischer Textanalyseverfahren.

Neben diesen Verfahren zur automatischen Textanalyse können computerunterstützte Inhaltsanalysen jedoch auch andere Dienstprogramme für die Eingabe und Verwaltung von Daten, und Bearbeitung elektronischer Texte eingesetzt werden.

Ausgehend von dieser Definition computerunterstützter und Inhaltsanalysen (siehe Abbildung 4) wird in den folgenden Kapiteln ein Methodeninventar automatischer Verfahren aufgebaut, das die Unterstützung kommunikationswissenschaftlicher Inhaltsanalysen verwendet werden kann. Verfahren, die bereits durch mich oder durch andere Forscher publiziert wurden, wird in verkürzter Form verwiesen, um den Verfahren, die nur in dieser Rahmenschrift diskutiert werden mehr Platz einräumen zu können.

Kapitel 4. Computerunterstützte Planungsphase

In der Planungsphase können Computer vor allem im Aufgabenbereich des Forschungsgegenstands, also bei der Planung der Textakquise und der Bestimmung relevanter und irrelevanter Unterthemen für die Inhaltsanalyse eingesetzt werden. Weiter empfiehlt es sich, die Planung und Hochrechnung des Aufwandes der Inhaltsanalyse durch Computer berechnen zu lassen. Für die Planung des Erhebungsinstrumentes, die grösstenteils aus definitorischer Vorbereitung und der Entscheidung für bestimmte Verfahren liegt, sind automatische Verfahren hingegen kaum hilfreich.

4.1. Textauswahl

Falls sich das Forschungsinteresse auf ein Thema bezieht, das über mehrere Teildiskurse und einen langen Zeitraum untersucht werden soll, so ist die Definition der relevanten Unterthemen, Medienangebote und Zeiträume, sowie die Ziehung einer analysierbaren und repräsentativen Stichprobe des Teilkorpus eine Herausforderung für den Forscher. Erschwert wird diese Aufgabe in Projekten, in denen die Berichterstattung über mehrere Länder oder über einen Zeitraum mehrerer Jahre oder Jahrzehnte verglichen wird, was zum einen eine Multiplikation der Anzahl Texte (Schäfer, Ivanova, & Schmidt, 2011), zum anderen auch eine Ausweitung der diskutierten Unterthemen zur Folge haben kann (vgl. Kirilenko, Stepchenkova, Romsdahl, & Mattis, 2012). Diese Herausforderungen stellen sich insbesondere bei der Analyse öffentlicher Debatten zu latenten Themen wie Arbeitslosigkeit oder Umweltschutz oder bei einem Vergleich mehrerer Wahlen oder Volksabstimmungen, wo mit einer intensiven Berichterstattung, thematischer Heterogenität und einer Durchdringung des gesamten Mediensystems zu rechnen ist.

Durch eine Stichprobenziehung kann das relevante Teilkorpus entlang dreier Dimensionen reduziert werden. Erstens kann eine Medienstichprobe gezogen werden, bei der die Berichterstattung in wenigen Medienangeboten stellvertretend für die Berichterstattung im Land erhoben wird. Häufig werden hier nationale und überregionale Leitmedien (z.B: Schäfer et al., 2011) verwendet. Falls die Inhaltsanalyse jedoch anschliessend mit Befragungsdaten kombiniert werden sollen, empfiehlt sich eine breitere Auswahl reichweitenstarker Medien (vgl. Schemer, Wirth, & Matthes, 2012; Wettstein, 2012a).

Zweitens kann eine zeitliche Stichprobe gezogen werden, um nicht den gesamten Verlauf der Debatte erfassen zu müssen. Eine solche zeitliche Stichprobe kann über eine künstliche Woche (vgl. Rössler, 2010: 59), eine Auswahl mehrerer Zeiträume mit einem festen Abstand (z.B. Avraham,

Wolfsfeld, & Aburaiya, 2000) oder eine gezielte Auswahl Ereignissen und deren Berichterstattung gezogen werden (vgl. Hanke, 2003).

Drittens kann eine Zufallsstichprobe aller Texte gezogen werden, um eine repräsentative Stichprobe der gesamten Berichterstattung zu erheben.

Die Entscheidungen, die in diesem Arbeitsschritt getroffen werden, haben weitreichende Auswirkungen für die Inhaltsanalyse. Um hier informierte Entscheidungen für Art und Umfang des relevanten Teilkorpus ziehen zu können, bieten sich qualitative oder explorative Voranalysen an, anhand derer der Umfang und Verlauf der Debatte umrissen werden kann. Die Voranalysen können aus Interviews mit Journalisten und Experten (vgl. Hajer, 2003: 282) bestehen oder aus qualitativen Inhaltsanalysen weniger ausgewählter Texte (vgl. Gerhards, 2003). Auch explorative automatische Inhaltsanalyen (Kirilenko et al., 2012) können zu diesem Zweck verwendet werden, um die relevanten Teildiskurse und die bedeutsamen Phasen im Verlauf der Berichterstattung zu identifizieren.

In diesem Abschnitt werden zwei automatische und ein computerunterstütztes Verfahren vorgestellt, die solche Voranalysen in kurzer Zeit erlauben. Sie ermöglichen mit wenig Aufwand einen gründlichen Überblick über die zu untersuchenden Themenbereiche und mögliche wichtige Ereignisse, die im Erhebungszeitraum stattgefunden haben und einen Einfluss auf die Erhebung haben könnten.

4.1.1. Term-Mapping zur Bestimmung der Unterthemen

Ein erstes Verfahren, das sich vor allem in der Frühphase der Planung relevanter Teilkorpora eignet, ist das *Term-Mapping* (vgl. Wettstein, 2012b). Dieses untrainierte, statistische Bag-of-Words Verfahren hilft in zwei Schritten bei der Identifikation und Interpretation zentraler Schlüsselworte innerhalb einer Debatte. In einem ersten Schritt werden Begriffe gesucht, die eine hohe Wahrscheinlichkeit haben, in einem Textkorpus gemeinsam mit einem oder zwei zentralen Begriffen der Debatte aufzutreten. In einem zweiten Schritt wird das gemeinsame Auftreten der Begriffe mit Hilfe einer Smallest Space Analyse (SSA: vgl. Guttman, 1968) ausgewertet und grafisch dargestellt. Das Ergebnis einer SSA lässt sich intuitiv und ohne Vorkenntnisse (vgl. Bloombaum, 1970) qualitativ interpretieren und gewährt einen Überblick über Unterthemen der Berichterstattung.

Hintergrund

Textstatistische Analysen basieren auf der Annahme, dass das Vokabular, mit dem Diskurse geführt werden, sich je nach Aspekt (z.B: Argumentationen, Unterthemen, Positionen, Frames) in geringfügig unterscheidet. Einzelne Texte, die auf einen bestimmten Aspekt des Diskurses

fokussieren, lassen sich daher anhand der verwendeten Begriffe induktiv unterscheiden. So zeigte Gerhards (2003), dass sich die Konfliktparteien in der Abtreibungsdebatte durch ihren exklusiven Gebrauch des Wortes ***Baby*** oder ***Fötus*** voneinander abheben. Miller und Riechert (2001) konnten zudem zeigen, dass die Schlüsselworte einer bestimmten Argumentationslinie in einer Debatte meist gemeinsam in Texten auftreten und sich nur selten mit den Begrifflichkeiten anderer Positionen vermischen.

Dieser Umstand erlaubt es, durch eine statistische Auswertung des gemeinsamen Auftretens von Begriffen in Dokumenten, die einzelnen Aspekte heterogener Debatten und ihre jeweils zentralen Begriffe induktiv zu bestimmen. Die Ansätze, die für diese Art der automatischen Inhaltsanalyse angewandt werden, können zwei unterschiedliche Ansätze unterschieden werden. Bei der Kontingenzanalyse von Schlüsselbegriffen wird zunächst eine Liste zentraler Schlüsselbegriffe definiert, deren gemeinsames Auftreten bestimmt und mittels multidimensionaler Skalierungsverfahren (z.B: Clusteranalyse, Faktoranalyse, SSA) ausgewertet wird (z.B: Miller & Riechert, 2001; Conway, 2006; Crawley, 2007). Bei explorativen textstatistischen Verfahren wie der *Latent Dirichlet Allocation* (LDA: vgl. Blei, Ng & Jordan, 2003) oder anderen *Concept Mapping*-Verfahren (vgl. Popping, 2000: 55; Quinn, Monroe, Colaresi, Crespin, & Radev, 2010) werden alle vorkommenden Begriffe in die Analyse einbezogen um induktiv jene Gruppen von Begriffen zu finden, deren gehäuftes gemeinsames Auftreten auf eine semantische Verbindung hinweist. Das Ergebnis kann schliesslich vor dem Hintergrund des Diskurses qualitativ ausgewertet und auf inhaltliche Aspekte der Debatte zurückgeführt werden (vgl. Gerhards, 2003: 306).

Das Term-Mapping stellt mit seinen zwei Schritten einen Kompromiss zwischen diesen beiden Verfahren dar. In einem ersten Schritt werden alle Begriffe verwendet, um induktiv eine kurze Liste von Schlüsselbegriffen zu definieren, die für eine Debatte zentral sind. In einem zweiten Schritt wird diese kurze Liste verwendet, um mittels einer SSA eine interpretierbare Darstellung der einzelnen Unterthemen der Debatte zu generieren (vgl. Wettstein, 2012b). Da beide Schritte automatisch durchgeführt werden, geht diese Analyse sehr schnell und kann innert Minuten einen Überblick über die Debatte und die zentralen Begriffe liefern.

Vorgehen[1]

Für die Durchführung einer Term-Mapping Analyse wird ein Textkorpus von ungefähr 10^4 Texten benötigt, das repräsentativ für die Berichterstattung der untersuchten Region und Zeit ist. Je nach Anzahl der untersuchten Medienangebote und Ausgaben kann eine Stichprobe der Berichterstattung gezogen werden, um das Korpus zu reduzieren. Das Korpus sollte insbesondere vor der Durchführung in flexionsreichen Sprachen so vorbereitet werden, dass Worte durch ihre jeweiligen Wortstämme repräsentiert werden[2]. In einigen Fällen bietet es sich auch an, wichtige Synonyme zu Gruppen zusammenzufassen, so dass mehrere gleichbedeutende Begriffe durch einen Platzhalter repräsentiert werden. Ein Beispiel für eine solche Repräsentation ist die Zusammenfassung der Begriffe ***Kanzlerin***, ***Bundeskanzlerin***, ***CDU-Vorsitzende*** und ***Angela Merkel*** zum Begriff ***Merkel*** (vgl. Miller, Andsager, & Riechert, 1998).

Im ersten Schritt des Term-Mapping wird das gemeinsame Auftreten aller Worte (*w*) mit einem vordefinierten zentralen Begriff (*z*) der Debatte bestimmt. Das Verfahren kann mehrfach für verschiedene zentrale Begriffe durchgeführt werden, falls sich dies für die eine Debatte anbietet. Das gemeinsame Auftreten ist dabei über die bedingten Wahrscheinlichkeiten definiert, dass w in Texten auftritt, in welchen *z* vorkommt ($P_{(w|z)}$) und dass *z* in Texten mit w auftritt ($P_{(z|w)}$). Begriffe, welche eine hohe Wahrscheinlichkeit haben, in Artikeln mit dem zentralen Begriff aufzutreten, jedoch selber nur selten vom zentralen Begriff begleitet werden ($P_{(w|z)} >> P_{(z|w)}$), sind grösstenteils Pronomen, Präpositionen oder feste formale Bestandteile der Texte (z.B.: "Autor: "). Umgekehrt sind seltene Begriffe, die hauptsächlich gemeinsam mit dem zentralen Begriff auftreten und sonst nur schwach verbreitet sind ($P_{(w|z)} << P_{(z|w)}$), sehr spezifisch für die Debatte und interessant für das Term-Mapping. Durch die Berechnung aller bedingten Wahrscheinlichkeiten lässt sich eine Rangliste der wichtigsten Begriffe für ein Thema generieren.

Für den zweiten Schritt einer Term-Mapping Analyse werden 50-100 der gefundenen wichtigen Begriffe ausgewählt. Dazu gehört auch der zentrale Begriff selber. Für diese Begriffe wird nun paarweise das gemeinsame Auftreten im Textkorpus berechnet, um daraus eine Distanzmatrix über alle Begriffe zu erstellen. Diese Matrix kann durch eine SSA skaliert und in einem zwei- oder

---

[1] Eine ausführlichere Beschreibung des Verfahrens und seiner Anwendung kann meinem Tagungsbandbeitrag zu Term-Mapping (2012b) entnommen werden (siehe Anhang A).

[2] Programme, mit denen diese Vorarbeit geleistet werden können, werden als Lemmatizer oder Stemmer bezeichnet und sind für viele Sprachen online frei verfügbar. Eine umfangreiche Sammlung dieser Programme wird beispielsweise unter http://snowball.tartarus.org/ angeboten. Dort sind auch die Algorithmen im Hintergrund der Stemmer dokumentiert.

dreidimensional in einem Scatterplot dargestellt werden. Das Ergebnis ist eine graphische Darstellung der wichtigsten Begriffe einer Debatte, in welcher die häufig gemeinsam auftretenden Begriffe optisch erkennbare Cluster bilden. Je nach Form der Debatte kann die resultierende Darstellung einzelne klar definierte Gruppen von Begriffen beinhalten, die für Unterthemen mit jeweils eigenem Wortschatz stehen, oder eine diffuse Wolke von Begriffen, die sich nur schwer in Gruppen einteilen lässt (siehe Abbildung 5).

Abbildung 5: Mögliche Formen von Debatten. a) Einzelnes zentrales Unterthema; b) Zwei getrennte Unterthemen; c) Vier eng verbundene Unterthemen. Quelle: Wettstein, 2012b.

Das Ergebnis des Term-Mapping ist für die Planung und Vorbereitung der folgenden Schritte der Inhaltsanalyse in mehrfacher Hinsicht von Bedeutung. Erstens weist das Verfahren auf mögliche Unterthemen und Teildiskurse hin, über deren Einbezug oder Ausschluss bei der Planung des relevanten Teilkorpus entschieden werden muss. Zweitens liefert das Verfahren eine Liste von Begriffen, die für die vorliegende Debatte bedeutsam sind und für das Auffinden von Texten in der Entwicklungsphase verwendet werden können. Drittens erlaubt die intuitiv interpretierbare Darstellung der Debatte als Scatterplot eine erste qualitative Analyse des Medieninhalts, was bei der Entwicklung von Kategorien und Themenlisten hilfreich sein kann.

Für die konkrete Anwendung kann das hier beschriebene Verfahren in unterschiedlichen Programmiersprachen implementiert und ausgeführt werden. In *Aeglos* (siehe Anhang B) ist Term-Mapping implementiert und erfordert als Eingabe lediglich eine Sammlung von Texten oder ein bestehendes Korpus. Durch vorbereitete Stemming-Algorithmen in mehreren Sprachen ist keine Vorbereitung der Korpora nötig. Als Ausgabe wird ein R-Skript generiert, mit dem abschliessend eine SSA gerechnet und dargestellt werden kann. Durch diese Verschränkung von Textanalyse- und Statistikprogrammen ist es für Anwender auch ohne vertiefte Kenntnisse möglich, eine Term-Mapping Analyse innert weniger Minuten durchzuführen.

4.1.2. Longitudinale Analyse zur Bestimmung von Kommunikationsereignissen

Bei Inhaltsanalysen langfristig aktueller Themen, die über einen ausgedehnten Zeitraum untersucht werden sollen, stellt sich bei der Definition des relevanten Teilkorpus die Frage nach konkreten Zeiträumen, in denen die Berichterstattung untersucht werden soll. Dies ist insbesondere der Fall bei öffentlichen Debatten, die über mehrere Jahrzehnte aktuell sind. Diese beginnen "irgendwo in der Vergangenheit und setzen sich in eine Zukunft fort, deren zeitliche Ausdehnung unbestimmbar ist" (Schwab-Trapp, 2003: 175). In dieser Zeitspanne wird zudem nicht konstant über die Debatte berichtet, sondern es ereignen sich unregelmässige Aufmerksamkeitsschübe, welche die Routineberichterstattung durchbrechen und dem Thema für kurze Zeit breite Beachtung verschaffen (vgl. Kepplinger, 2001: S. 123; Leskovec, Backstrom, & Kleinberg, 2009).

Diese kurzen Aufmerksamkeitsschübe können durch Ereignisse, politische Debatten oder Medienkampagnen ausgelöst werden (vgl. Wettstein, In Press) und sind durch die thematische Aufarbeitung und die Beteiligung unterschiedlicher Akteure am Diskurs, die sie jeweils mit sich bringen (vgl. Kepplinger & Habermeier, 1995; Zhu, 1992), für eine Inhaltsanalyse besonders interessant. Da je nach Forschungsfrage die Beschreibung der Berichterstattung als Reaktion auf wichtige Ereignisse, die politische Einflussnahme auf den Diskurs oder die Berichterstattung über politische Debatten im Fokus stehen kann, ist es für die Planung der zu erhebenden Zeiträume hilfreich, bereits vor einer quantitativen Inhaltsanalyse den Verlauf einer Debatte in einem bestimmten Zeitraum beschreiben und zwischen unterschiedlichen Auslösern für erhöhte Medienaufmerksamkeit unterscheiden zu können.

In diesem Zusammenhang entwickelte ich (In Press) ein Verfahren, mit dem es möglich ist, aufgrund des Verlaufs der Fokussierung der Berichterstattung auf die Ursachen für plötzliche Aufmerksamkeitsschübe zu schliessen. Dieses Verfahren, welches sich nach einem ersten Test anhand quantitativer Inhaltsanalysedaten als vielversprechend erwiesen hat, kann theoretisch auch mit Daten einer oberflächlichen automatischen Inhaltsanalyse durchgeführt werden (vgl. Wettstein, in press). Es würde damit bereits im Vorfeld einer quantitativen Inhaltsanalyse eine Beschreibung des Verlaufs und eine Identifikation von Ereignissen und politischen Debatten ermöglichen.

Vorgehen

Für die Unterscheidung von Aufmerksamkeitsschüben, die durch Ereignisse, Kampagnen oder politische Debatten ausgelöst wurden, ist eine Erfassung des zeitlichen Verlaufs des Berichterstattungsvolumens, sowie der Fokussierung auf Akteure und Unterthemen nötig. Diese drei Dimensionen des Verlaufs einer Debatte lassen Rückschlüsse auf den Ursprung von Aufmerksamkeitsschüben zu (vgl. Wettstein, In Press). Für eine automatische Analyse des Verlaufs

wird deswegen eine computergestützte Messung des Volumens und der Vielfalt bezüglich Themen und Akteuren für jeden Tag der Berichterstattung benötigt.

Während das Volumen über die Anzahl und Länge veröffentlichter Beiträge zu einem Thema relativ leicht erhoben werden kann, erfordert eine Bestimmung der Verteilung von Akteuren und Themen in einer Debatte weiterführende Textanalyseverfahren. Für die Bestimmung unterschiedlicher Akteure bietet sich eine automatische Bestimmung von Akteuren über Named Entity Recognition (vgl. Wüest et al., 2011) oder eine syntaktische Analyse zur Identifikation von Sprechern (vgl. van Atteveldt 2015) an, um die Akteure in jedem Beitrag zu identifizieren. Da für diese Auswertung lediglich die Vielfalt oder Fokussierung der Akteure relevant ist, ist eine exakte und verlässliche Bestimmung von Akteuren keine zwingende Voraussetzung.

Die thematische Vielfalt an einzelnen Tagen kann mit unterschiedlich präzisen Methoden erfasst werden. Sehr schnell aber unpräzise ist die Suche nach einer Liste von Schlüsselworten der Debatte, die auf unterschiedliche Unterthemen oder Teildiskurse hinweisen. Eine Fokussierung auf einzelne Begriffe kann als thematische Fokussierung gewertet werden. Verlässlichere Methoden wären in diesem Zusammenhang jedoch textstatistische Verfahren zur induktiven Bestimmung von Unterthemen (vgl. Blei et al., 2003), deren Auftreten anschliessend in einzelnen Texten nachgewiesen werden kann.

Liegt für jeden Tag eine Bestimmung des Volumens und der vorkommenden Themen und Akteuren vor, so kann mit Hilfe der von Boydstun (2008: 181) vorgeschlagenen Berechnung der Verdichtung der Berichterstattung die thematische und akteursbezogene Fokussierung berechnet werden:

$$Fokus = 1 - \frac{-\sum_i^N p_i * \ln(p_i)}{\ln(N)}$$

Hierfür wird für jeden Tag der Anteil *pi* einzelner Akteure und Themen an der Berichterstattung bestimmt und mit der Gesamtanzahl (*N*) potentiell vorkommender Akteure oder Themen in Beziehung gesetzt. Das Resultat ist ein Koeffizient für die Fokussierung, der den Wert 0 erreicht, falls alle Elemente gleichmässig verteilt sind und den Wert 1, falls die Berichterstattung nur auf einen Akteur oder ein Thema beschränkt ist.

In meinem Beitrag (In Press) beschreibe ich fünf unterschiedliche Typen von Aufmerksamkeitsschüben, die sich bezüglich der Verläufe von Themen- und Akteursfokussierung unterscheiden lassen. Um diese Typen zu unterscheiden, wird in einem ersten Schritt nach Spitzenwerten des Volumens der Berichterstattung gesucht. Anschliessend wird nach lokalen Minima und Maxima der Fokussierung in der Nähe dieser Aufmerksamkeitsschübe gesucht. Je nach Zeitpunkt

und Richtung der Fokussierung auf Themen und Akteure kann dann auf unterschiedliche Ursachen für den Aufmerksamkeitsschub geschlossen werden (siehe Abbildung 6).

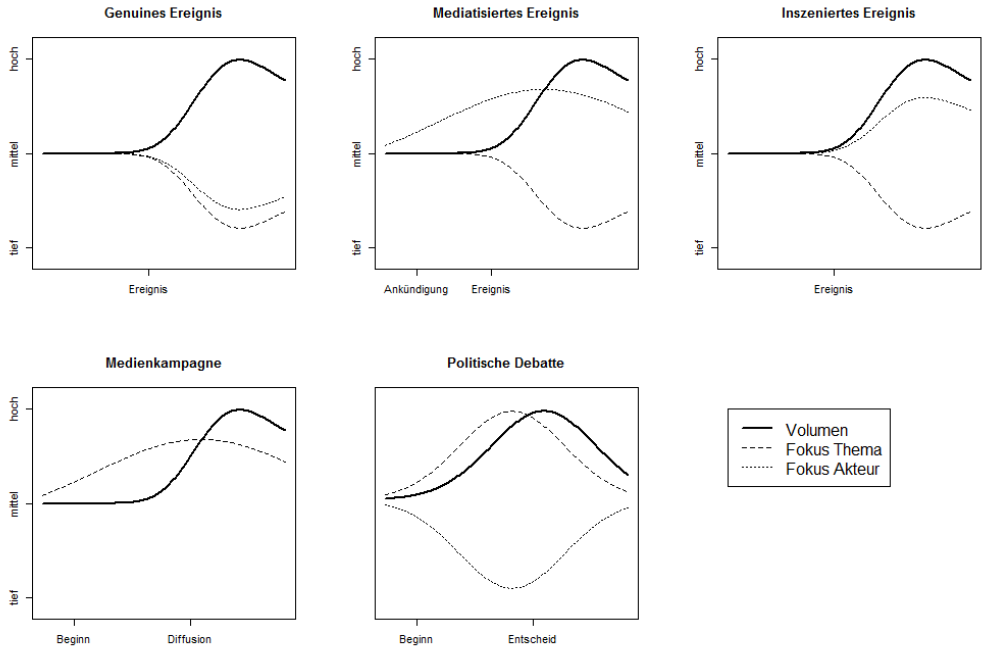


Abbildung 6: Hypothetischer Verlauf des Volumens und der Fokussierung der Berichterstattung vor und während Aufmerksamkeitsschüben mit unterschiedlichen Auslösern. Quelle: Wettstein, In Press.

So weist eine thematische Fokussierung vor einem Aufmerksamkeitsschub darauf hin, dass ein Unterthema bereits vor der erhöhten Medienaufmerksamkeit intensiver beachtet wurde. Dies macht eine sich zuspitzende politische Debatte oder eine Kampagne als Auslöser für die Medienaufmerksamkeit wahrscheinlich. Diese beiden Auslöser lassen sich über die Akteursvielfalt unterscheiden, die im Fall politischer Debatten sehr hoch ist, im Fall gezielter Kampagnen durch einen Akteur jedoch eher nicht.

Das Ergebnis dieser Analyse ist eine Beschreibung der möglichen Ursachen für jeden Aufmerksamkeitsschub in der Debatte, die keine qualitative Inhaltsanalyse der Berichterstattung erfordert. Es empfiehlt sich dennoch, gerade für die Zeiträume, die für die Analyse in Betracht gezogen werden, eine qualitative Bestätigung der vermuteten Auslöser über die Inspektion der Titelseiten einer beliebigen Zeitung zum Beginn der einzelnen Aufmerksamkeitsschübe.

4.1.3. Optimieren der Auswahl von Schlüsselworten zur Textakquise

In der Planung der Textakquise sind neben der Definition relevanter Themen und Unterthemen auch die Suchkriterien für Texte vorzubereiten. Zunächst gilt es dabei, konkrete Bedingungen festzulegen, unter welchen ein Text für die Inhaltsanalyse relevant ist und an welchen Kriterien dies erkannt werden kann. Handelt es sich bei den Texten um audiovisuelles Material oder um analog

vorliegende Texte, so sind dies die endgültigen Kriterien, anhand welcher ein Mitarbeiter oder Codierer relevante Texte erkennen und auswählen kann. Werden in der Inhaltsanalyse auch elektronisch vorliegende Texte aus Online-Medien, Textdatenbanken oder anderen digitalen Medien analysiert, so beinhaltet die Vorbereitung der Suchkriterien auch eine Sammlung von Suchbegriffen, anhand welcher relevante Texte gefunden werden können. Hier kann ein computerunterstütztes Verfahren helfen, um die Suchbegriffe einzugrenzen.

Hintergrund

Der Aufbau einer Liste von Suchbegriffen kann, gerade bei breiten Debatten mit mehreren Unterthemen und Problemfeldern anspruchsvoll sein, da es gilt, eine Liste zu erstellen, welche möglichst sparsam, vollständig und präzis für die jeweilige Suchanfrage ist. *Sparsamkeit* meint in diesem Zusammenhang, dass die Liste so kurz wie möglich sein sollte, um sie auch in Fällen anwenden zu können, in denen keine langen Suchstrings möglich oder sinnvoll sind. Als *vollständig* kann eine Liste angesehen werden, wenn anhand der Begriffe alle relevanten Texte gefunden werden können und keine weiteren Suchbegriffe benötigt werden. *Präzision* weist schliesslich auf die Eigenschaft der Suchbegriffe hin, hauptsächlich relevante Texte zu finden und einen möglichst geringen Anteil falscher Treffer zu generieren.

Da die drei Kriterien sich widersprechen können, sollte ihre relative Wichtigkeit vor der Analyse festgelegt werden. So kann die Sparsamkeit hinter die Vollständigkeit und Präzision gestellt werden, wenn alle Texte aus Quellen stammen, die lange Suchbegriffe zulassen. Falls in der Entwicklungsphase trainierte automatische Verfahren eingesetzt werden, um relevante von irrelevanten Texten zu trennen, so ist auch die Präzision deutlich weniger relevant als die Vollständigkeit der Suchbegriffe.

Um eine Liste von Suchbegriffen zu erstellen, die alle drei Kriterien berücksichtigt, kann ein computerunterstütztes Verfahren eingesetzt werden, das ohne den Einsatz spezialisierter Software über Suchmasken beliebiger Textdatenbanken durchgeführt werden kann. Das mehrstufige Verfahren geht dabei von einer anfänglichen Liste aus, die über eine qualitative, computerunterstützte Analyse so lange verfeinert wird, bis alle drei Kriterien möglichst gut erfüllt sind. In diesem Abschnitt wird das Vorgehen schrittweise erläutert, um es in Kapitel 4.1.4 an einem realen Beispiel zu demonstrieren.

Vorgehen

Die Suche nach Schlüsselbegriffen beginnt jeweils mit einer qualitativen Analyse der Debatte, die entweder manuell über die Lektüre von Texten zum Debattenthema, über Experteninterviews oder über automatische Verfahren wie das Term Mapping (siehe Kapitel 4.1.1) vorgenommen werden

kann. In den meisten Fällen empfiehlt sich eine Kombination aller drei Methoden. Aus dieser ersten Analyse wird ersichtlich, welche Unterthemen berücksichtigt werden sollten und welche Worte in relevanten Texten vorkommen können. Das Ergebnis dieses ersten Schritts ist eine möglicherweise noch lange Liste von Begriffen, die für den Diskurs unterschiedlich zentral sind und unterschiedliche Unterthemen abdecken.

In einem zweiten Schritt ist eine möglichst exakte Definition relevanter Texte nötig, um sie in den folgenden Schritten von irrelevanten Texten unterscheiden zu können. Hier kann man sich auf die zuvor vorgenommene qualitative Analyse stützen, um gegebenenfalls einzelne Unterthemen (z.B: Auslandsberichterstattung) auszuschliessen oder häufig vorkommende irrelevante Texte (z.B: Fallbeispiele und Einzelschicksale) explizit auszunehmen. Anhand der festgelegten Kriterien sollten unterschiedliche Personen zuverlässig zwischen relevanten und irrelevanten Texten unterscheiden können.

Der dritte Schritt der Suche nach Schlüsselworten besteht in einer Zusammenstellung einer kurzen Liste möglichst zentraler Begriffe aus der anfänglichen Sammlung von Begriffen. Hierbei kann ausgenutzt werden, dass einzelne Suchsysteme Wildcard-Symbole kennen (z.B. * oder !), die für jede beliebige Abfolge von Zeichen am Ende, am Anfang oder in der Mitte eines Worts stehen können. Gerade bei flexionsreichen Sprachen wie Deutsch sind diese Symbole wichtig, um mit einem Suchbegriff mehrere Begriffe abzudecken (z.B: schliesst ***Arbeitslos**** unter anderem auch ***Arbeitslosigkeit***, ***Arbeitslosenquote***, ***Arbeitslosenversicherung***, ***Arbeitslose*** ein). Ebenso können einzelne Suchmaschinen feste Ausdrücke aus mehreren Worten oder Worte in einem bestimmten Kontext finden. Auch diese Möglichkeiten können in diesem Schritt ausgeschöpft werden. Das Ergebnis dieses dritten Schrittes ist eine anfängliche Liste von Suchbegriffen, die weder sparsam noch präzis oder vollständig sein muss.

Im vierten Schritt wird anfängliche Liste von Suchbegriffen computerunterstützt auf Eignung überprüft. Dazu wird eine Suchanfrage aus der Liste erstellt und für die Suche nach Texten in einer Datenbank verwendet, welche komplexe Suchanfragen mit booleschen Operatoren zulässt. Die Anzahl der aufgefundenen Artikel ist für die weiteren Schritte zu notieren. Im Anschluss wird nach den Texten gesucht, die jeder einzelne Begriff zum Suchergebnis beiträgt, indem eine Suchanfrage mit Ausschlussbedingung formuliert wird, in welcher dieser Begriff (Testbegriff) gegen alle anderen Begriffe in der Liste (Begriff1-Begriff N) gestellt wird:

Testbegriff NOT (Begriff1, Begriff2...., Begriff N)

Diese Suchanfrage liefert nur jene Artikel, die ausschliesslich bei Einbezug des Testbegriffes gefunden werden. Durch die Wiederholung dieser Prozedur für alle Suchbegriffe der anfänglichen Liste entsteht für jeden Begriff ein eigenes Suchergebnis. Je nach Art des Zugriffes auf die Datenbank

können die Suchanfragen automatisch generiert und die Suchergebnisse abgespeichert werden, was den Aufwand dieser Prozedur deutlich verringert.

In einem fünften Schritt werden die Ergebnisse für die einzelnen Begriffe manuell auf die Anzahl Treffer und den Anteil relevanter Texte innerhalb der Suchergebnisse überprüft. Da hier nur eine grobe Schätzung vorgenommen werden muss, reicht für diesen Zweck eine oberflächliche Prüfung der ersten 20-50 Suchergebnisse für jeden Suchbegriff. Steuert ein Begriff dem Suchergebnis ausschliesslich irrelevante Texte bei, so kann dieser zur Entfernung aus der Liste vorgemerkt werden. Steuert ein Begriff hauptsächlich irrelevante Texte aber auch einige relevanten zu, so sollten die wenigen relevanten Texte auf Worte untersucht werden, die sie von den irrelevanten Texten abheben, um diese als mögliche weitere Suchbegriffe aufzunehmen und den ursprünglichen Begriff zu ersetzen.

Sind alle Begriffe geprüft, können die Begriffe, die wenige relevante Texte beisteuern, entfernt und gegebenenfalls durch präzisere Suchbegriffe ersetzt werden. Die resultierende Liste sollte erneut getestet werden, bis die anfängliche Liste möglichst sparsam und präzis ist. Das Ergebnis dieses Schrittes ist eine Liste von Begriffen, die für die Suche nach relevanten Texten geeignet ist, jedoch noch nicht vollständig, da möglicherweise wichtige Begriffe fehlen.

Nun kann die Liste durch weitere Begriffe erweitert werden. Dazu werden Worte, die in der qualitativen Analyse ebenfalls als mögliche Schlüsselworte identifiziert wurden, auf ihren Mehrwert in der Suche nach Texten geprüft. Jedes zusätzliche Wort wird erneut gegen die zuvor gefundene Suchanfrage getestet, indem nur nach Artikeln gesucht wird, die der zusätzliche Begriff beisteuern kann. Auch hier wird eine Abschätzung gemacht, inwiefern einzelne Begriffe bei der Suche nach relevanten Texten helfen können. Durch ein Überfliegen neu gefundener relevanter Texte können zudem weitere Suchbegriffe gefunden werden. Am Ende dieses Schrittes werden alle Suchbegriffe der Suchanfrage hinzugefügt, die das Ergebnis um zusätzliche relevante Texte erweitern. Die Liste von Suchbegriffen wird damit vollständiger, jedoch büsst sie an Sparsamkeit und Präzision ein.

Je nach Gewichtung der drei Eigenschaften, welche die Suchbegriffe erfüllen sollen, kann im Anschluss der vierte Schritt wiederholt und die Liste wieder eingeschränkt werden. Auch andere Schritte innerhalb des Prozesses können am Ende wiederholt werden, der Aufwand bei der manuellen Prüfung der Ergebnisse ist dabei aber kritisch dem Nutzen entgegenzustellen, der durch zusätzliche Analyseschritte entsteht. Das Ergebnis dieses Verfahrens ist eine Suchanfrage, die in der Entwicklungsphase das Auffinden relevanter Texte in Archiven oder Datenbanken erlaubt.

4.1.4. Anwendungsbeispiel: Debatte zur Arbeitsmarktpolitik

Um die Machbarkeit und den Nutzen der drei hier vorgestellten Verfahren zu einer computerunterstützten Planung des relevanten Textkorpus zu demonstrieren, wird eine explorative Vorstudie zu einer quantitativen Inhaltsanalyse der Arbeitsmarktdebatte in der Schweiz, Deutschland und Österreich im Frühjahr 2014 durchgeführt. Die quantitative Inhaltsanalyse sollte dabei möglichst alle Aspekte und Teildiskurse der Debatte, inklusive spezifischer nationaler Problemfelder der drei Länder, umfassen. Um dies zu gewährleisten, wurden drei computerunterstützte Vorstudien durchgeführt, die einen Überblick über die Debatte in den drei Ländern ermöglichen sollten.

Bestimmung der Suchbegriffe

In der quantitativen Inhaltsanalyse sollten mehrere Medienangebote in jedem Land erfasst werden, die über Datenbanken und Abonnemente in digitaler Form verfügbar sind. Der erste Schritt in der Planung der Inhaltsanalyse war deswegen die Zusammenstellung von Schlüsselworten, mit welchen später relevante Texte für die Analyse gefunden werden können. Relevante Texte waren in dieser Studie all jene Texte, die sich mit einem Arbeitsrechtlichen oder Arbeitsmarktpolitischen Problembereich oder mit Arbeitslosigkeit oder Erwerbstätigkeit im Inland beschäftigen. Auslandsberichterstattung, Einzelschicksale und Texte, in denen nur die Erwerbstätigkeit einer Person erwähnt wird, sollten nicht Teil des Korpus werden.

Um mögliche Suchbegriffe zu finden, wurden zunächst politikinteressierte Studierende aus allen drei Ländern nach aktuellen Problemen und Diskussionen im Zusammenhang mit Arbeitsmarktpolitik und Arbeitslosigkeit befragt. Zudem wurde in allen drei Ländern ein Term-Mapping mit jeweils 2000 Texten durchgeführt, um Begriffe im Zusammenhang mit dem Kernbegriff *Arbeits** zu finden. Auf diese Weise konnten folgende mögliche Unterthemen in den drei Ländern ausgemacht werden: Lohnpolitik und Boni, Arbeitslosigkeit und Erwerbsquote, Sicherheit am Arbeitsplatz, Öffnungs- und Arbeitszeiten, Unternehmensregulierung und Wirtschaftsförderung.

Aus dieser anfänglichen Beschreibung der Debatte wurde eine erste Liste von Schlüsselbegriffen definiert, die für alle drei Länder identisch war. Gleichzeitig wurde für jedes Land eine zusätzliche Liste von möglicherweise relevanten Begriffen der jeweils spezifischen nationalen Problemfelder definiert, die ebenfalls als Suchbegriffe in Frage kommen. Drei Mitglieder der Projektleitung sollten diese Listen mit Hilfe des oben beschriebenen computergestützten Verfahrens evaluieren und so weit reduzieren, dass eine vollständige und sparsame Liste von Suchbegriffen für jedes Land vorliegt. Die Präzision sollte als zweitrangig betrachtet werden, da in der Entwicklungsphase ein trainiertes Verfahren die relevanten von den irrelevanten Texten trennen würde.

Für die Durchführung des Verfahrens wurde die zentrale Liste in eine Suchanfrage für die Textdatenbank LexisNexis übersetzt (siehe Tabelle 2), um dort nach Texten im März und April 2014 zu suchen. Die Suchergebnisse wurden so gespeichert, dass der Titel und der Kontext der gefundenen Suchergebnisse sichtbar waren. Der Kontext der Suchbegriffe wurde jeweils überflogen, um den Bezug der ersten 40 gefundenen Texte zum Thema der einheimischen Arbeitsmarktpolitik zu bestimmen. Die Codierer schätzten auf einer vierstufigen Skala ein, wie gross der Anteil relevanter Texte in diesem Suchergebnis ist (keine Texte, einzelne Texte, viele Texte oder meiste Texte). Die Ergebnisse dieser Suche sind in Tabelle 2 aufgeführt. Kleine Unterschiede in der Relevanz einzelner Begriffe (z.B. ***Rentenalter*** oder ***Mindestl****) führten zu drei leicht unterschiedlichen zentralen Suchanfragen für Arbeitsmarktpolitik in den drei Ländern.

Tabelle 3: Bestimmung des Beitrags zentraler Begriffe der Suchanfrage zum Auffinden relevanter Texte in drei Ländern. Angegeben sind für jedes Land und jeden Suchbegriff jeweils die Anzahl Texte, welche dieser Begriff zum Suchergebnis beiträgt und die Relevanz dieser Begriffe auf einer 4-stufigen Skala (keine, einzelne, viele, meist).

Begriff	Deutschland (Spiegel, Focus, Berliner Zeitung)	Österreich (Die Presse, Der Standard)	Schweiz (Tagesanzeiger, Weltwoche)
gewerkschaft!	138 Texte, viele relevant	141 Texte, viele relevant	43 Texte, einzelne relevant
!arbeitslos!	148 Texte, meist relevant	143 Texte, einzelne relevant	69 Texte, einzelne relevant
mindestl!	47 Texte, meist relevant	14 Texte, einzelne relevant	19 Texte, meist relevant
arbeitgeberverb!	2 irrelevante Texte	0 Texte	1 irrelevanter Text
wirtschaftsverb!	8 irrelevante Texte	4 irrelevante Texte	8 Texte, 1 relevant
arbeitspl!	161 Texte, viele relevant	175 Texte, meist relevant	95 Texte, viele relevant
sozialabbau	1 irrelevanter Text	1 irrelevanter Text	1 irrelevanter Text
arbeitsmarkt!	38 Texte, meist relevant	46 Texte, viele relevant	24 Texte, einzelne relevant
rentenalter	6 Texte, einzelne relevant	1 irrelevanter Text	6 Texte, einzelne relevant
Ergebnis:	gewerkschaft! OR !arbeitslos! OR mindestl! OR arbeitspl! OR arbeitsmarkt! OR rentenalter	gewerkschaft! OR !arbeitslos! OR mindestl! OR arbeitspl! OR arbeitsmarkt!	gewerkschaft! OR !arbeitslos! OR mindestl! OR wirtschaftsverb! OR arbeitspl! OR arbeitsmarkt! OR rentenalter

Die zentralen Suchanfragen für jedes Land wurden anschliessend verwendet, um die Eignung weiterer Begriffe zu testen. Hierfür wurden in den einzelnen Ländern unterschiedliche Begriffe getestet, die sich in der Voranalyse als mögliche Kandidaten herausgestellt hatten. Das Ergebnis dieser Analyse ist in Tabelle 3 aufgeführt und führte wiederum zu länderspezifischen Anpassungen der Suchanfrage.

Das Endergebnis der computerunterstützten Suche nach Schlüsselworten waren in diesem Fall drei unterschiedliche, auf Vollständigkeit optimierte Suchanfragen, die speziell auf Deutschland, die Schweiz und Österreich zugeschnitten waren. Die Sparsamkeit der Suchbegriffe stand durch die Suche in Online-Datenbanken nicht im Vordergrund. Auch auf die Präzision wurde im Hinblick auf

eine spätere gezielte Auswahl relevanter Texte weniger Wert gelegt als auf ein vollständiges Suchergebnis, in dem möglichst wenige relevante Texte fehlen.

Tabelle 4: Bestimmung des Beitrags weiterer Suchbegriffe zum Auffinden relevanter Texte in drei Ländern. Angegeben sind für jedes Land und jeden Suchbegriff jeweils die Anzahl Texte, welche dieser Begriff zum Suchergebnis beiträgt und die Relevanz dieser Begriffe auf einer 4-stufigen Skala (keine, einzelne, viele, meist).

Begriff	Deutschland (Spiegel, Focus, Berliner Zeitung)	Österreich (Die Presse, Der Standard)	Schweiz (Tagesanzeiger, Weltwoche)
streik!	49 Texte, einzelne relevant	45 Texte, keine relevant	10 Texte, einzelne relevant
pensionsalter	0 Texte	13 Texte, viele relevant	1 irrelevanter Text
hartz!	32 Texte, viele relevant	-	-
DGB	7 irrelevante Texte	-	-
SGB	-	-	0 Texte
ÖGB	-	18 Texte, einzelne relevant	-
arbeitnehm!	66 Texte, viele relevant	99 Texte, viele relevant	16 Texte, einzelne relevant
jobcenter	12 irrelevante Texte	-	-
boni	7 Texte, einige relevant	14 Texte, einzelne relevant	18 Texte, einzelne relevant

Der aufwändigste Schritt in dieser Analyse ist die qualitative Voranalyse, aus welcher Ideen für mögliche Suchbegriffe gewonnen werden. Dieser Vorgang dauerte für mehrere Personen zwei Tage. Die computerunterstützte Optimierung der Suchanfrage konnte in einer bis zwei Stunden pro Land durchgeführt werden und stellte sich damit als effiziente Methode für die Planung der Textakquise heraus.

Beschreibung der Debatte

Auch wenn die unterschiedlichen Suchanfragen bereits erste Rückschlüsse auf die Inhalte der Arbeitsmarktdebatte in den drei Ländern zuliessen, sollte die Debatte in den drei Ländern im Zeitraum zwischen März und Juni 2014 etwas detaillierter beschrieben werden, um ein besseres Bild der einzelnen Teildiskurse und möglicher bedeutsamer Ereignisse zu erhalten. Für diesen Zweck wurde eine explorative Analyse der Aufmerksamkeitsschübe und ein Term-Mapping durchgeführt. In diesem Abschnitt wird das Ergebnis dieser Analyse am Beispiel der Schweiz berichtet.

Für eine Beschreibung der Arbeitsmarktdebatte in der Schweiz zwischen März und Juni 2014 wurden mit Hilfe der oben beschriebenen Suchanfrage alle Artikel in diesem Zeitraum im *Blick* (*N*=891), der *Neuen Zürcher Zeitung* (*N*=1983), dem *Tages-Anzeiger* (*N*=1332) und der *Weltwoche* (*N*=403) aus diversen Datenbanken beschafft. Die Texte wurden lokal gespeichert und ihr Erstelldatum wurde automatisch erfasst, um später eine Zeitreihenanalyse durchführen zu können.

Die erste Analyse der Texte bestand in einem Term-Mapping über den gesamten Zeitraum mit den zentralen Begriffen ***arbeitnehm**** und ***arbeitslos****. Für die Darstellung wurden aus den 150 Begriffen, die am häufigsten mit den beiden Begriffen genannt wurden, jene ausgesucht, welche in

weniger als 50% der Texte auftraten und für welche die Bedingung galt: P(z|w) >> P(w|z). Damit wurden vor allem Pronomen und Präpositionen aussortiert und es blieben 34 zentrale Begriffe für die graphische Darstellung (siehe Abbildung 7).

Abbildung 7: Ergebnis des Term-Mapping in der Schweiz zwischen dem 1.3. und 30.6.2014. Zentrale Begriffe: arbeitslos und arbeitnehm.

Das Term-Mapping erlaubt einen ersten Überblick über die Debatte, die sich durch zwei dichte Cluster von Begriffen auszeichnet, die einerseits mit Mindestlöhnen und der Mindestlohninitiative zusammenhängen (links) und andererseits mit Unternehmen, Arbeitnehmern und Arbeitgebern (mitte). Gleichzeitig zeigt sich auf der rechten Seite der Darstellung eine eher lockere Sammlung von Begriffen, die mit der Ausbildung und der Situation am Arbeitsplatz in Zusammenhang stehen.

Für eine Beschreibung des Verlaufs der Debatte wurde eine Analyse der Fokussierung auf Themen und Akteure mit dem geringstmöglichen Aufwand durchgeführt. Um möglichst schnelle Ergebnisse zu erzielen wurde die Anzahl Themen und Akteure, welche an jedem Tag genannt wurden, über das Auftreten von Schlüsselbegriffen operationalisiert. Als thematische Begriffe dienten dabei die 32 Schlüsselworte aus dem Term-Mapping, die keinen Akteur bezeichneten. Als Akteure wurden die grössten Parteien (***svp***, ***sp***, ***fdp***, ***gruen***, ***bdp***, ***glp***, ***gruenliberal***), sowie die Begriffe ***gewerkschaft*** und ***sgb*** aus dem Term-Mapping verwendet. Für diese insgesamt 41 Begriffe wurde mit Hilfe von *Aeglos* eine Term-Document Matrix (TDM) erstellt, aus der anschliessend mit Hilfe von *Nogrod* Verlaufskurven mit einem gleitenden Fenster von sieben Tagen berechnet wurde. Die Anzahl der

Nennungen der 41 Begriffe an jedem Tag diente dabei als Indikator für das Volumen der Berichterstattung.

Die resultierenden Verlaufskurven wurden graphisch dargestellt und optisch auf lokale Maxima der Berichterstattung untersucht (Abbildung 8). Für alle fünf identifizierten Aufmerksamkeitsschübe wurde das Muster der Fokussierung auf Themen und Akteure im Vorfeld und während des Schubes ebenfalls optisch identifiziert und mit den idealtypischen Verläufen abgeglichen, um die Art des Aufmerksamkeitsschubes zu ermitteln.

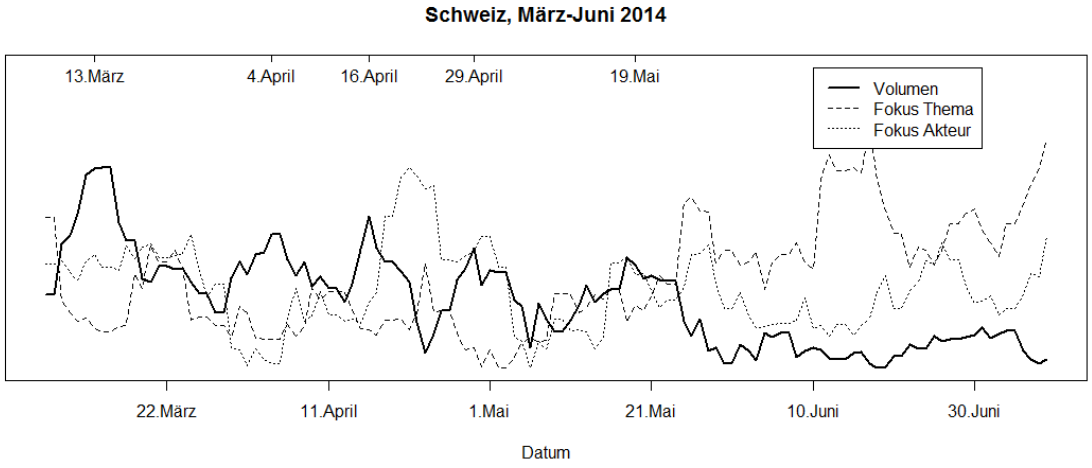


Abbildung 8: Umfang und Fokussierung der Berichterstattung in der Schweiz zwischen März und Juni 2014. Die Werte wurden an ihrer empirischen Spannweite standardisiert. Die Daten, an welchen die Berichterstattung ein lokales Maximum erreicht, sind über den Kurven angegeben.

Für den ersten Aufmerksamkeitsschub, der am 13. März 2014 seinen Höhepunkt erreicht, konnte ein gleichzeitiges Minimum der thematischen Fokussierung ausgemacht werden. Dieses Muster lässt auf ein unvorhergesehenes Ereignis schliessen, das sich in diesen Tagen zugetragen hat. Das gleiche Muster ist auch für den Aufmerksamkeitsschub mit Höhepunkt am 4. April zu beobachten, der nicht nur eine hohe thematische Vielfalt, sondern auch eine hohe Akteursvielfalt aufweist.

Vor dem dritten Aufmerksamkeitsschub am 16. April verstärkt sich sowohl die Fokussierung auf Themen als auch auf Akteure, was auf eine Kampagne hinweist. Dieses Muster wiederholt sich erneut vor dem nächsten Aufmerksamkeitsschub am 29. April. Das fünfte Maximum des Berichterstattungsvolumens wird schliesslich am 19. Mai nach einer stetig ansteigenden Berichterstattung mit einer gleichzeitig stetig ansteigenden thematischen Fokussierung und einer hohen Akteursvielfalt erreicht. Dieses Muster weist auf eine politische Debatte hin, die am 19. Mai ihren Höhepunkt erreicht.

Um die Natur und die Inhalte der einzelnen Aufmerksamkeitsschübe zu überprüfen, wurden gezielt einzelne Titelseiten der Zeitung 20Minuten (die nicht Teil dieser Analyse war) konsultiert. Es

wurden jeweils jene Ausgaben gewählt, die zu Beginn der Anstiegs der Berichterstattung veröffentlicht wurden, um die Ursachen der Aufmerksamkeitsschübe zu ermitteln. Die beiden Ereignisse, welche die ersten beiden Aufmerksamkeitsschübe ausgelöst hatten, waren nach dieser inhaltlichen Prüfung die am 12. März veröffentlichte Ankündigung der Post, tausende von Stellen abzubauen und die am 31. März angekündigte Einführung eines Mindestlohns bei H&M.

Die politische Debatte, die am 19. Mai ihren Höhepunkt erreichte, war die Abstimmung über die Einführung eines Mindestlohns in der Schweiz am Sonntag, dem 18.Mai. Die beiden Kampagnen, die vor dieser Abstimmung ausgemacht wurden, waren Abstimmungskampagnen, die fünf Wochen und drei Wochen vor der Abstimmung intensiv geführt wurden. Die auffällig stark reduzierte Berichterstattung über Arbeitsmarktthemen im Juni lässt sich indes auf die Fussball-WM 2014 zurückführen, welche ab Juni die Medienagenda dominierte.

Wie schon die computerunterstützte Suche nach Schlüsselbegriffen stellte sich auch die explorative Beschreibung der Berichterstattung als ein sehr effizientes Verfahren heraus, das insgesamt nur wenige Stunden in Anspruch nahm. Der zeitintensivste Schritt bei den beiden angewandten Verfahren ist die automatisch durchgeführte Analyse des gemeinsamen Auftretens von Begriffen, die durch die Vorbereitung der Texte und die Zusammenstellung von Häufigkeits- und Kreuztabellen fast eine Stunde Rechenzeit benötigte. Die graphische Darstellung, die optische Identifikation der Aufmerksamkeitsschübe und Verlaufsmuster, sowie die Überprüfung der Ergebnisse anhand einer zusätzlichen Zeitung nahmen nur jeweils wenige Minuten in Anspruch.

Fazit

Die drei hier vorgestellten und beispielhaft durchgeführten Verfahren zur Unterstützung der Planungsphase gewähren einen schnellen und differenzierten Überblick über die zu untersuchende Berichterstattung, die zentralen Begriffe in den einzelnen Artikeln und einschneidende Ereignisse und politischen Debatten, die in der weiteren Planung berücksichtigt werden müssen. Mit der optimierten Suchanfrage und einem Überblick über mögliche Teildiskurse liefern diese Verfahren wichtige Ergebnisse für die Textakquise und die konkrete Formulierung von Kategorien und Beispielen für das Codebuch. Sie stehen damit inhaltlich einer qualitativen Vorstudie und Zusammenfassung der Berichterstattung in nichts nach, sind jedoch mit wenigen Stunden Zeitaufwand ausserordentlich effizient.

4.2. Projektplanung

Im Aufgabenbereich der Administration steht in der Planungsphase eine konkrete Projektplanung an, die neben einem konkreten Zeitplan auch eine Abschätzung der Anzahl Texte, Stunden und benötigten Mitarbeiter beinhaltet. Während diese Aufgabe nicht durch automatische Verfahren

ersetzt werden kann, so können Computer zumindest eine Unterstützung bei der konkreten Planung darstellen. Einerseits durch den Einsatz von Projektplanungssoftware, mit der ein Zeitplan für die einzelnen Arbeitsschritte aufgestellt werden kann, andererseits mit der vorausschauenden Anlage einer Datenbank für die Inhaltsanalyse. Die Tabellen dieser Datenbank sollten Aufstellungen der einzelnen Textquellen, Erhebungszeiträume, Codierer, Texte, Arbeitszeiten und Kostenpunkte enthalten. Dies erlaubt bereits in der Planungsphase die konkrete Abschätzung des Umfangs der Inhaltsanalyse und wird in der laufenden Inhaltsanalyse die konstante Überwachung des Fortschritts und der Kosten ermöglichen.

In den meisten Inhaltsanalyse-Projekten ist die technische Umsetzung einer solchen Datenbank über eine Arbeitsmappe in kommerzieller Tabellenkalkulationssoftware (z.B.: Excel, Open Calc) möglich und kann mit Hilfe von Formeln und benutzerdefinierten Abfragen und Grafiken einfach bearbeitet und gelesen werden. In besonders umfangreichen Projekten, in denen die Anzahl der Texte oder der weiteren eingebundenen Informationen so gross ist, dass eine Arbeitsmappe nicht mehr effizient verwendet werden kann, ist die Umsetzung als SQL-Datenbank ratsam.

4.2.1. Datenbanklösung für ein Grossprojekt

Die Inhaltsanalyse des NCCR-Moduls *"Populismus in Zeiten der Globalisierung und Mediatisierung"* (vgl. NCCR 2015). in deren Rahmen ein Teil der hier vorgestellten Verfahren entwickelt und angewandt wurde, soll in diesem Kapitel als Beispielstudie für den Aufbau einer Datenbank herangezogen werden. Da die Inhaltsanalyse die Berichterstattung in 11 Ländern über ein Jahr, sowie jeweils einige Wochen vor den vergangenen fünf Wahlen in acht dieser Länder untersucht, war bereits die Organisation der Zeiträume, Medien und Sprachen zu komplex, um sie ohne Datenbank zu lösen. Zudem wurde bereits zu Beginn mit einem Bedarf an mehr als zwanzig Codierern gerechnet, was ebenfalls eine gründliche Planung und übersichtliche Buchführung erforderte. Um die Daten des Projekts zu organisieren, wurde deswegen eine Datenbanklösung, bestehend aus einer umfassenden Excel-Arbeitsmappe und zwei zusätzlichen Tabellen zur Verwaltung der Texte und des Fortschritts in den einzelnen Teilanalysen erstellt. Im Folgenden werden die einzelnen Tabellen und ihre Inhalte in kurzen Zügen beschrieben.

Codierer: In einer Tabelle wurden die Informationen zu allen Codierern und sonstigen Mitarbeitern abgelegt. Diese umfassten neben dem Namen und dem internen Zeichen der einzelnen Codierer[3] auch ihre Kontaktinformationen, Sprachkenntnisse, das vereinbarte Arbeitspensum und die

[3] Um den Spagat zwischen der Anonymisierung eingegebener Daten und Codiererinformationen und der Nachvollziehbarkeit der Codierungen zu bewältigen, kann jedem Codierer ein internes Zeichen (Zahlen oder

Anstellungsdauer. Vor der Rekrutierung echter Codierer waren in der Tabelle lediglich Platzhalter mit bestimmten Sprachkenntnissen und realistischem Arbeitspensum eingetragen, um die verfügbare Arbeitskraft und die Anzahl benötigter Codierer abschätzen zu können. Die Platzhalter wurden dann in der Entwicklungsphase durch die tatsächlich verfügbaren Codierer ersetzt. Zudem wurde als weitere Information der Name des Verzeichnisses eingetragen, in welchem jeder Codierer seine Arbeit erledigt und seine Daten speichert.

Quellen: In einer Tabelle wurden alle Informationen zu den Quellen abgelegt, aus welchen die Texte für die Inhaltsanalyse stammen sollten. Da es sich dabei um Zeitungen, Zeitschriften, TV-Sendungen, Parteiprogramme und Pressemitteilungen aus unterschiedlichen Ländern handelte, gehörte zu den wichtigsten Informationen das Land, die Sprache und der Typ der Quelle. Gleichzeitig wurden auch die Codes eingetragen, welche die Medien in den Daten der Inhaltsanalyse erhalten sollten. Auch der Zugang zu den einzelnen Quellen, die Namen der Datenbanken und Websites, sowie die benötigten Passworte für jedes Abonnement wurden hier gespeichert. Schliesslich wurde in diese Tabelle auch die Anzahl zu codierender Artikel in jedem der zu untersuchenden Teilkorpora eingetragen. Auch hier wurden in der Planungsphase zunächst möglichst realistische Schätzungen verwendet, um die zu leistende Arbeit in allen Sprachen abschätzen zu können. Mit Beginn der Materialsammlung konnten die Informationen durch reale Zahlen ersetzt werden.

Teilkorpora: Eine Tabelle enthielt eine Übersicht über alle Teilkorpora, die in der Analyse untersucht werden sollten. Ein Teilkorpus umfasst dabei jeweils mehrere Quellen, einen Zeitraum und eigene Kriterien für die Relevanz von Texten. Diese Aufstellung ist spätestens im Arbeitsschritt der Datenaufbereitung von Bedeutung, wenn die unterschiedlichen Teile der Analyse wieder voneinander getrennt werden müssen. Für jedes Teilkorpus wurden Zeitraum, Sprache, Land, Teil der Inhaltsanalyse und der angestrebte Stichprobenumfang in die Tabelle eingetragen.

Aufwand: Eine Tabelle, die aufgrund der Informationen in der Codierer- und Quellen-Tabelle automatisch erstellt wurde, beinhaltete die Anzahl Artikel, den zu erwartenden Arbeitsaufwand, die verfügbare Arbeitskraft und die Abschätzung der Kosten für die einzelnen Teile der Inhaltsanalyse. Diese Tabelle war vor allem in der Planung und bei der konstanten Überwachung des Fortschritts von Bedeutung, um einen Überblick über den bereits geleisteten und bevorstehenden finanziellen und personellen Aufwand zu erhalten. Durch die Umsetzung der Datenbank in einer Excel-Arbeitsmappe konnte diese Tabelle über Formelfelder automatisch aktuell gehalten werden. Bei anderen Umsetzungen empfiehlt sich für eine solche Übersicht eine benutzerdefinierte Abfrage.

Buchstaben) zugewiesen werden, mit dem in der Erhebung alle Codierungen gekennzeichnet werden. Auf diese Weise lässt sich die Arbeit einzelner Codierer nachverfolgen, wenn man die Zuweisung der Zeichen kennt.

Beschaffung: Eine Tabelle wurde verwendet, um in der Entwicklungsphase den Fortschritt der Textakquise zu dokumentieren und die Anzahl gefundener Texte für jede Quelle und jeden Zeitraum der Erhebung zu verzeichnen. Aus dieser Tabelle schöpfte die Quellen-Tabelle jeweils die Informationen über die Grösse der Grundgesamtheit der Texte, aus der eine Stichprobe gezogen werden muss.

Texte: Ebenfalls in der Entwicklungsphase wurde eine Tabelle angelegt, die den Namen, den Speicherort, die Sprache und den Bearbeitungsstatus aller vorbereiteten Texte enthielt. Diese Tabelle ist insbesondere in der Anwendungsphase von Bedeutung, da sie jederzeit Auskunft über die Verfügbarkeit von Texten unterschiedlicher Stichproben gibt. Sie kann auch dafür verwendet werden, einzelnen Codierern neue Texte zuzuweisen.

Codierfortschritt: Für die Erhebungsphase wurde eine Tabelle angelegt, um laufend den Codierfortschritt der einzelnen Quellen in jedem Teilkorpus überwachen zu können. Wie schon bei der Kalkulation des Aufwands kann diese Tabelle auch als benutzerdefinierte Abfrage der Datenbank konzipiert werden. Basierend auf der Anzahl der Texte in der Grundgesamtheit jedes Mediums in jedem Teilkorpus kann in dieser Tabelle sowohl die Anzahl als auch der Anteil codierter Texte nachgeschlagen werden. Im Arbeitsschritt der Datenaufbereitung können aus diesen Daten Gewichtungsfaktoren für die einzelnen Quellen in den jeweiligen Teilkorpora berechnet werden, um zufällige Unterschiede in den Stichprobengrössen auszugleichen.

Daten: Ein integraler Bestandteil der administrativen Datenbank einer Inhaltsanalyse sind die erhobenen Daten. Diese können in vielen Inhaltsanalysen als einzelne Tabellen eingebunden werden, in Inhaltsanalysen mit einer relationalen Erhebung können Sie jedoch auch eine eigene Datenbank darstellen. Zumindest die Tabelle der bereits codierten Texte sollte in die administrative Datenbank eingebunden werden, um den Fortschritt überwachen zu können.

Arbeitszeiten: Eine letzte Tabelle enthielt schliesslich eine Aufstellung der geleisteten Arbeitszeit jedes Codierers in jedem Monat der laufenden Inhaltsanalyse. Diese Aufstellung ist zum einen für die Berechnung der Löhne, auf der anderen Seite für die Überprüfung der die Erfüllung vereinbarter Leistungen zentral. Auch hier kann die Tabelle entweder dynamisch in die Datenbank eingebunden oder als Abfrage konzipiert werden.

Insgesamt entstand mit dieser Datenbank die Basis für die administrative Verwaltung, Betreuung und kurzfristige Planung während der Inhaltsanalyse (siehe Abbildung 9). Durch die zahlreichen gegenseitigen Abhängigkeiten sind die Konzeption und der Aufbau einer solchen Datenbank unter Umständen eine Herausforderung. Für den deutlich geringeren administrativen Aufwand während

der Inhaltsanalyse und einen schnellen Einblick in den Fortschritt der Inhaltsanalyse lohnt sich diese Investition jedoch bei grossen Projekten.

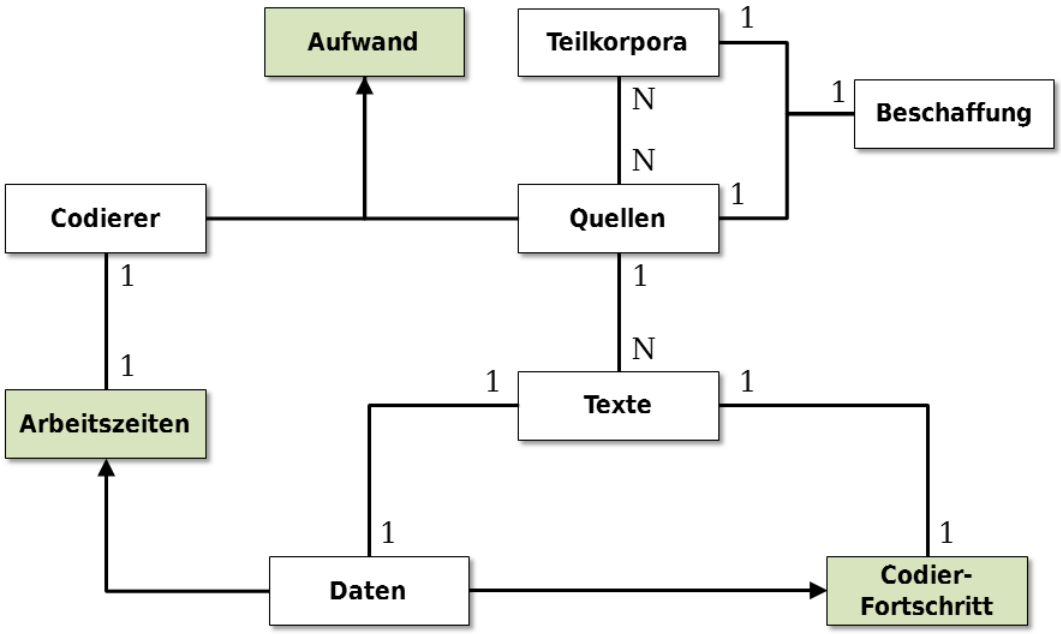


Abbildung 9: Datenstruktur der administrativen Planung der NCCR-Inhaltsanalyse
Anmerkungen: Grün hinterlegte Felder sind dynamische Tabellen oder Abfragen, welche automatisch generiert werden. Pfeile signalisieren die Tabellen, aus welchen die Informationen für die Berechnung stammen. Beschriftete Kanten geben an, ob die Zuweisung von Fällen in den einzelnen Tabellen eineindeutig (1-1) oder mehrfach (1-N; N-N) ist. Lesebeispiel: Jeder Text gehört zu genau einer Quelle. Eine Quelle enthält jedoch mehrere Texte.

Die Notwendigkeit und genaue Ausführung einer solchen Datenbank muss für jede Inhaltsanalyse separat beurteilt und geplant werden[4]. In jedem Fall empfiehlt sich jedoch hier der Einsatz von Computerprogrammen zur Verwaltung und Darstellung, da eine Buchhaltung auf Papierform keine dynamischen Verknüpfungen und keine automatischen Hochrechnungen erlaubt. Gerade diese sind für die Planung und Durchführung einer Inhaltsanalyse aber besonders hilfreich.

[4] In dem hier vorgestellten Projekt war die Datenbank mit 308 Teilkorpora mit insgesamt 250'000 Texten, von welchen 50'000 durch 87 Codierer bearbeitet wurden, nicht nur gerechtfertigt sondern dringend nötig, um jederzeit einen Überblick zu behalten.

Kapitel 5. Computerunterstützte Entwicklungsphase

In der Entwicklungsphase einer Inhaltsanalyse sind die Entwicklung des Erhebungsinstruments und die Sammlung von Codiermaterial zu leisten. Bei der Entwicklung eines Codebuches sind automatische Verfahren dabei nur von marginalem Nutzen, beispielsweise für die Suche nach Beispielen für einzelne Ausprägungen über eine Volltextsuche. Die Suche und Aufbereitung von Texten für das Textkorpus der Inhaltsanalyse kann jedoch an mehreren Stellen durch geeignete automatische Verfahren erleichtert werden. In diesem Kapitel wird deswegen in erste Linie auf diesen Arbeitsschritt eingegangen, für den automatische Verfahren bereits breite Anerkennung gefunden haben.

5.1. Textakquise

Die Durchführung der Textakquise kann in Inhaltsanalysen, welche Texte aus unterschiedlichen Quellen und in unterschiedlicher Form erfassen, mit hohem Aufwand verbunden sein. Zwar wird dieser Arbeitsschritt heute nur selten manuell durch eine systematische Suche in Zeitungsarchiven bewältigt, sondern mittels automatischer Suche von Texten in elektronischen Archiven und Online-Angeboten, auch diese Form der Textakquise kann jedoch zusätzlichen Aufwand bedeuten. Gerade bei der Analyse heterogener Themenbereiche, für welche eine Vielzahl von Suchbegriffen definiert werden muss, um die Texte vollständig zu erheben, führen Volltextsuchen oft zu grossen Mengen irrelevanter Texte, die einen der Suchbegriffe in einem themenfremden Kontext verwenden.

Die Ergebnisse müssen daher im Anschluss an die Volltextsuche bereinigt werden, um die irrelevanten Texte aus der Analyse auszuschliessen. Hier kann entweder manuell verfahren werden, was auch für geübte Codierer pro Text einen Aufwand mehreren Sekunden pro Text bedeutet, oder mit Hilfe automatischer Verfahren, welche die Texte auf ihre Relevanz hin überprüfen.

5.1.1. Zugänge

Der Forschung steht heute eine Vielzahl von Möglichkeiten offen, auf digitale Massenkommunikation zuzugreifen, um sie nach Schlüsselworten zu durchsuchen, lokal zu speichern und inhaltsanalytisch zu erfassen. Zu diesen Zugängen gehören Datenbanken von Pressetexten (z.B: Nexis, LexisNexis, Factiva), welche sowohl für die akademische als auch wirtschaftliche Nutzung konzipiert sind. Diese Datenbanken bieten neben der manuellen Suche und Speicherung auch Programmschnittstellen (API: application programming interface) für automatische Zugriffe auf ihre Archive an.

Texte, welche als Kurzmeldungen auf sozialen Netzwerken (z.B: Twitter, Facebook) veröffentlicht werden, können ebenfalls über speziell eingerichtete Programmschnittstellen gesucht und lokal

gespeichert werden. Programme, welche auf diese Schnittstellen zugreifen und Texte erfassen können, sind beispielsweise als Bibliotheken für R oder Python (Beispiele) verfügbar. Für einen schnellen Zugriff ohne Programmieraufwand kann das Open Source Programm Facepager (Keyling & Jünger, 2013) verwendet werden.

Ein weiterer Zugang, welcher sich vor allem für Pressetexte von Online-Medien anbietet, sind automatisch generierte Zusammenfassungen der Veröffentlichungen (RSS: rich site summary), über welche mittels geeigneter Programme auf einzelne Artikel zugegriffen werden kann (vgl. Trilling, 2014). In ähnlicher Weise können auch über Ankündigungen von Online-Medien auf Twitter und anderen sozialen Netzwerken einzelne Artikel gefunden werden. Für die lokale Speicherung und Analyse dieser Texte von den jeweiligen Websites sollte der Inhalt der Website jedoch von Werbung und zusätzlichen Informationen ausserhalb des Artikels befreit werden (vgl. Scharkow, 2012: 113; Günther & Scharkow, 2014). Auch hier bietet der *NewsClassifier* (Scharkow, 2015) vorgefertigte Funktionen an.

Bei Medienangeboten, welche weder in Datenbanken noch Online frei verfügbar sind, besteht teilweise die Möglichkeit, digitale Ausgaben in PDF-Format zu abonnieren. Auch hier können Texte mit geringem Aufwand elektronisch gesucht und extrahiert werden, um sie für eine Inhaltsanalyse zugänglich zu machen. Für diesen Zugang ist jedoch sowohl mit Abonnementskosten als auch mit einem kleinen Programmieraufwand zu rechnen, was diesen Zugang für die meisten Anwendungen nicht zur besten Option macht.

5.1.2. Maschinelles Lernen zur Auswahl relevanter Texte

Der breite Zugang zu digitalen Ausgaben von Medientexten erlaubt die Beschaffung und Speicherung einer grossen Zahl von Texten, auf die eine konkrete Suchanfrage zutrifft. Da gerade bei umfassenden Themen wie einer breiten öffentlichen Debatte oder einem Wahlkampf die Suchanfrage möglichst breit gefasst sein muss, um keine relevanten Texte auszulassen, besteht das Risiko, eine Vielzahl irrelevanter Texte zu erhalten. Diese sollten vor der Bearbeitung durch Codierer aussortiert werden, um keine Zeit mit der Codierung von nicht relevanten Texten zu verlieren.

Für die Identifikation relevanter oder irrelevanter Texte schlägt Scharkow (2012) eine trainierte automatische Klassifikation vor. Bei diesem Verfahren wird zunächst eine Stichprobe von ca. 100-200 Texten manuell klassifiziert, um anschliessend einen automatischen Klassifikationsalgorithmus zu trainieren, diese Entscheidungen an unbekannten Texten durchzuführen. Für das Training erfasst der Algorithmus die Begriffe, welche in Texten mit unterschiedlicher Klassifikation auftreten. Mittels statistischer Verfahren wird dann nach einem Muster im Auftreten von Begriffen gesucht, das eine Vorhersage der Gruppenzugehörigkeit von Texten ermöglicht. Durch die wiederholte Anwendung dieses Musters auf neue Texte und die manuelle Kontrolle der Klassifikation lernt das Programm

schliesslich, Texte unterschiedlicher Gruppen zu trennen (vgl. Scharkow, 2015). In ähnlicher Weise können Texte auch nach Themengebieten oder anderen inhaltlichen Kriterien klassifiziert werden (vgl. Scharkow, 2012: 202).

Im Zuge dieser Dissertation wurde ein ähnliches Verfahren in *Aeglos* implementiert, um einerseits die Relevanz von Texten für die Inhaltsanalyse zu bestimmen und andererseits eine trainierte halbautomatische Inhaltsanalyse (siehe Kapitel 7.1.2) zu realisieren. Das Programm verwendet eine naive Bays'sche Klassifikation, bei welcher für jeden Begriff im Trainingskorpus die Wahrscheinlichkeit berechnet wird, in Texten unterschiedlicher Klassen aufzutreten. Die Häufigkeit des Auftretens der einzelnen Begriffe in unbekannten Texten ermöglicht es anschliessend, die ungefähre Wahrscheinlichkeit zu berechnen mit welcher der Text zu jeder der trainierten Klassen gehört (vgl. Manning & Schütze, 1999: 596).

Das Ergebnis dieser Klassifikation ist für jeden Text und jede Klasse eine Zahl zwischen 0 und 1, welche angibt, mit welcher Wahrscheinlichkeit er zu dieser Klasse (z.B: Der Klasse relevanter Texte) gehört. Abhängig von der Güte und des Umfangs des Trainingsmaterials können diese Vorhersagen unterschiedlich zutreffend sein. Für die tatsächliche Auswahl relevanter Texte empfiehlt sich deswegen ein kritischer Test der Vorhersagen anhand eines ebenfalls manuell klassifizierten Testkorpus. Die Übereinstimmung der Vorhersage mit der tatsächlichen Klassifikation kann statistisch ausgewertet werden, um Richtwerte für die Arbeit mit dem Klassifikationsalgorithmus zu erhalten.

Anwendung im NCCR

Eine trainierte Klassifikation wurde in der Inhaltsanalyse des NCCR -Moduls "*Populismus in Zeiten der Globalisierung und Mediatisierung*" angewandt, um relevante von irrelevanten Texten unterscheiden zu können. Für diesen Zweck wurde zunächst ein Trainingskorpus von zwei Mitgliedern der Projektleitung erstellt, die für jede Sprache 500 Texte manuell klassifizierten. Die Klassifikation wurde mittels einer dichotomen Entscheidung in *Angrist* durchgeführt und dauerte pro Text im Durchschnitt 6.03 Sekunden (SD=7.51). Für irrelevante Texte wurde dabei etwas mehr Zeit (M=7.59; SD=8.77) benötigt als für relevante Texte (M=5.26; SD=6.67). Der Unterschied war signifikant ($F(1,1497)=32.69$; $p<.01$) und weist darauf hin, dass vor der Klassifikation eines Textes als irrelevant länger gezögert wurde. Das Trainingskorpus wurde verwendet, um in *Aeglos* die dichotome Kategorie der Relevanz zu trainieren und ein Testkorpus mit Texten aus der Schweiz (N=446), Deutschland (N=1880) und England (N=780) zu klassifizieren, die durch die Codierer während der Inhaltsanalyse bereits als relevant oder irrelevant eingestuft wurden.

Die Qualität der Klassifikation wurde über eine logistische Regression geprüft, indem die Zugehörigkeit zu relevanten oder irrelevanten Texten mit der automatisch berechneten Wahrscheinlichkeit vorhergesagt wurde. In Abbildung 10 sind die Ergebnisse dieser Auswertung für die Einschätzung der Relevanz von Texten dargestellt. An dieser Darstellung erkennt man, dass die Vorhersage zwar den relevanten Texten signifikant höhere Werte zuweist, die Trennung zwischen relevanten und irrelevanten Texten jedoch nicht eindeutig ist. Es gibt keinen Vorhersagewert, welcher die relevanten klar von den irrelevanten Texten trennt.

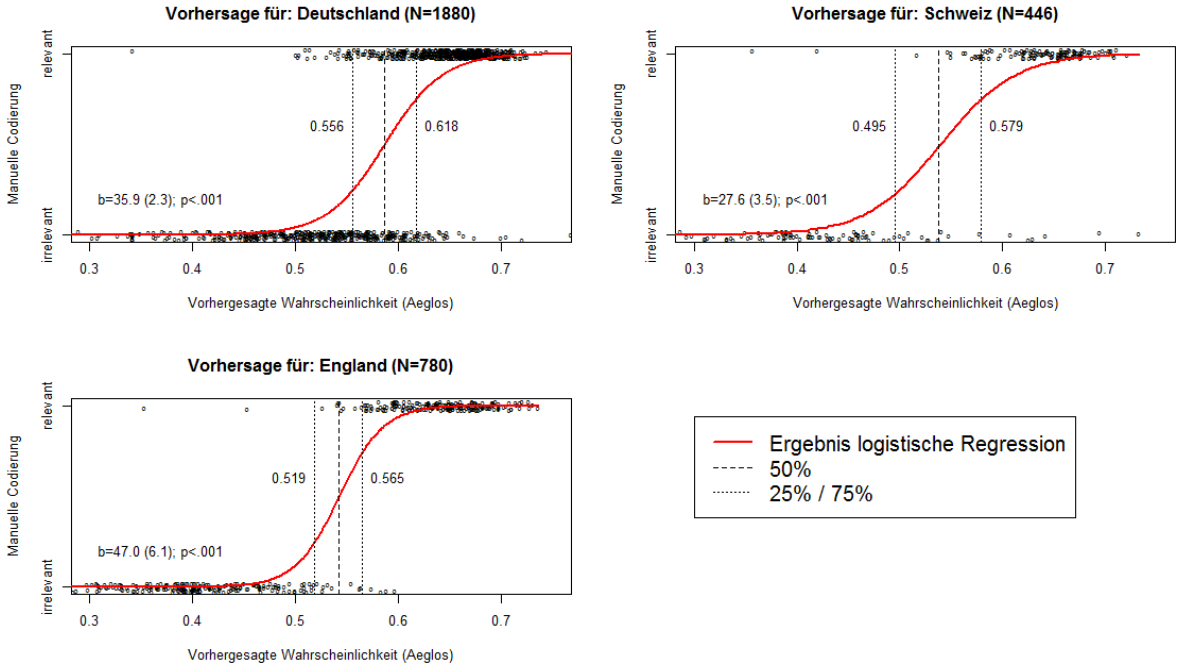


Abbildung 10: Ergebnisse der logistischen Regression zur Erklärung der Relevanz mittels automatischer Klassifikation.

Anmerkung: Die senkrechten Linien zeigen an, ab welcher automatisch vorhergesagten Wahrscheinlichkeit die mittels logistischer Regression berechnete Chance von 25%, 50% oder 75% besteht, dass ein Text relevant ist.

In solchen Fällen muss abgewogen werden, ob für die Analyse eher eine gewisse Anzahl irrelevanter Texte im Textkorpus oder der Verlust einiger relevanter Texte in Kauf genommen werden kann. Abhängig von der Entscheidung in dieser Frage kann die berechnete Wahrscheinlichkeit, ab welcher ein Text für relevant befunden wird, höher oder tiefer angesetzt werden. Die logistische Regression kann dabei helfen, diese Grenzen festzulegen. Im Beispiel, welches in Abbildung 10 visualisiert ist, wurde eine möglichst tiefe berechnete Wahrscheinlichkeit gewählt, um das Risiko zu minimieren, relevante Texte zu verlieren. Konkret wurde jeweils der Punkt gewählt, ab welchem aufgrund der logistischen Modellierung eine 25% Wahrscheinlichkeit besteht, dass ein Text tatsächlich relevant ist (linke gestrichelte Linien). Texte wurden damit für Deutschland ab einer vorhergesagten Wahrscheinlichkeit von 55.6% (CH: 49.5%; UK: 51.9%) als möglicherweise

relevante Texte in die Analyse aufgenommen. Damit konnte jeweils ungefähr die Hälfte der heruntergeladenen Texte vor der Herausgabe an Codierer als irrelevant verworfen werden und mussten nicht mehr während der Analyse bestimmt werden.

Um das Training weiter zu verbessern und auch in Grenzfällen eindeutiger zu machen, empfiehlt es sich, die bei der automatischen Vorhersage als knapp relevant oder irrelevant eingestuften Texte manuell auf Relevanz zu prüfen und dem Trainingskorpus hinzuzufügen. Je grösser die Anzahl korrekt klassifizierter Grenzfälle im Trainingskorpus ist, desto besser funktioniert auch in diesen Fällen die Vorhersage.

5.1.3. Vorbereitung des Textkorpus

Die finale Aufgabe im Arbeitsschritt der Textakquise ist die Vorbereitung des Textkorpus für die anstehende Inhaltsanalyse. Hierfür sollte zum einen eine Bestandsaufnahme aller Texte gemacht werden, um eine Stichprobe ziehen und alle Texte mit einer eindeutigen Identifikation zu versehen, zum anderen sollten die Texte formal für die Bearbeitung durch die Codierer vorbereitet werden.

Falls die Texte mittels einer geeigneten Software oder manuell von digitalen Archiven, Websites oder anderen elektronischen Quellen beschafft wurden, liegen sie in der Regel in einem Format vor, welches durch Dienstprogramme festgelegt wird. Die häufigsten Formate sind CSV-Tabellen, SQL-Datenbanken, XML- oder HTML-formatierte Dokumente, (Rich-)Textdateien und PDF-Dokumente. In den meisten Anwendungen werden dazu jeweils mehrere Texte in einer Datei abgelegt. Um die Texte für die Codierer zugänglich zu machen, sollten alle Texte in ein einheitliches und gut lesbares Format überführt werden, das bei der Erhebung verwendet werden kann.

Eine automatische Verarbeitung des Rohmaterials empfiehlt sich hier vor allem für grosse Inhaltsanalysen, ist jedoch nur selten mit Hilfe vorgefertigter Programme möglich. Zwar sind Bibliotheken für R oder Python vorhanden, mit denen sich Informationen aus XML- oder HTML-Dokumenten extrahieren lassen, dennoch ist die Extraktion von Texten meist mit zusätzlichem Programmieraufwand verbunden. Eine einfache Möglichkeit zur Trennung mehrerer Texte in einem Dokument wurde in *Aeglos* implementiert. Mit der Funktion 'Split collocated text files' lassen sich lange Sammlungen von Texten automatisch in durchlaufend nummerierte Textdateien für eine anschliessende Verteilung an Codierer umwandeln.

5.2. Erhebungsinstrument

Die Entwicklung eines quantitativen Codebuches oder eines Leitfadens für eine Inhaltsanalyse ist eine Arbeit, die nicht von automatischen Verfahren übernommen werden kann. Hier sind durch den Forscher Begriffe zu definieren, das Vorgehen festzulegen und Beispiele anzubringen. Automatische Verfahren können bei dieser Arbeit höchstens dabei helfen, über eine Volltextsuche einzelne reale

Beispiele im Textkorpus zu finden. Gerade im Hinblick auf parallele oder halbautomatische computerunterstützte Erhebungen gibt es in diesem Arbeitsschritt jedoch vorbereitende Arbeiten, die geleistet werden müssen.

5.2.1. Vorbereitung paralleler computergestützter Inhaltsanalyse

Bei der parallelen computergestützten Inhaltsanalyse werden einzelne Kategorien des Codebuchs automatisch erhoben und erst im Arbeitsschritt der Datenaufbereitung wieder mit den manuellen Daten verknüpft. Bei der Konzeption des Erhebungsinstrumentes kann bereits geprüft werden, inwiefern sich einzelne Kategorien auch automatisch erheben lassen. Für die automatischen Erhebungen muss in diesem Schritt ebenfalls ein Erhebungsinstrument – also ein Algorithmus zur Textanalyse – entwickelt und getestet werden. Eine solche parallele Umsetzung und Prüfung automatischer Verfahren bedeutet in der Entwicklungsphase einen Mehraufwand, kann sich jedoch durch Zeitersparnis in der Erhebungsphase auszahlen.

Kategorien, die sich besonders für eine parallele Analyse eignen, sind formale Aspekte, die bei einer manuellen Codierung einen unnötigen Aufwand für den Codierer bedeuten würden. So kann beispielsweise die Länge eines Textes automatisch ermittelt werden, ohne dass der Codierer tatsächlich Worte zählen muss. Ebenso lässt sich das Erstelldatum oder der Titel eines Textes oftmals automatisch auslesen und muss nicht durch den Codierer manuell abgetippt werden.

Zusätzlich zu diesen leicht messbaren Kategorien können auch kommerzielle oder frei verfügbare *Sentiment-Detection*-Programme zur Bestimmung des Tonus (z.B: Emotionalität, Negativität, Konfliktivität) eines Artikels anwenden (vgl. Hart, 2001; Young & Soroka, 2012). Diese Programme können parallel zu den menschlichen Codierern die Sprache von Texten analysieren und damit zusätzliche quantitative Daten für die Inhaltsanalyse liefern.

Schliesslich kann auch geprüft werden, ob sich einzelne Kategorien für eine automatische Analyse mittels trainierter Klassifikation eignen. Hierfür bietet der *NewsClassifier* (siehe Scharkow, 2012; 2015) leicht zu verwendende Funktionen an, um die automatische Vorhersage von Ausprägungen aufgrund eines Trainingskorpus mit manuell codierten Texten zu testen. Scharkow (2012) findet in ausführlichen Tests der Trainierbarkeit automatischer Verfahren, dass sich Themengebiete (Sport, Politik, Wirtschaft) relativ leicht und bereits mit einem Trainingskorpus von 300 Texten trainieren lassen (S. 202). Inhaltliche Kategorien wie das Vorkommen von Ereignissen, Kriminalität oder Wahlen sind jedoch schlecht trainierbar und eignen sich für diese Art der Erhebung nicht.

Prinzipiell sollte bei der Entwicklung paralleler automatischer Erhebungen jeweils geprüft werden, wie gross ihr Nutzen im Vergleich zum Entwicklungsaufwand steht. Dabei kann auch bedacht werden, dass die automatische Erhebung nicht simultan zur manuellen Inhaltsanalyse stattfinden muss,

sondern auch im Nachhinein noch möglich ist. Diese Aufgabe der Entwicklungsphase lässt sich also auch in die Anwendungsphase oder ans Ende der Inhaltsanalyse verschieben, wo gegebenenfalls mehr Ressourcen zur Verfügung stehen.

5.2.2. Vorbereitung halbautomatischer Inhaltsanalyse

Bei der halbautomatischen Inhaltsanalyse bearbeiten die Codierer die Texte in einer Eingabemaske (GUI: Graphical User Interface) am Computer. Im Hintergrund werden die Texte gleichzeitig automatisch analysiert, um den Codierer bei seiner Arbeit zu unterstützten und ihm Vorschläge zur Codierung zu unterbreiten (siehe Kapitel 7.1.2). Im Hinblick auf diese Art der Erhebung sind zwei Dinge zu beachten: Erstens sollte das Codebuch in einer Form erarbeitet werden, die eine Umsetzung als GUI erlaubt. Es sollte also bereits bei der Definition der Kategorien daran gedacht werden, was der Codierer für eine Auswahl zu treffen hat, wie er um eine Eingabe gebeten wird und wie die Daten gespeichert werden sollen. Zweitens sollte beachtet werden, dass die automatische Vorhersage von Codierungen besonders gut funktioniert, wenn kleinteilige, präzise Informationen aus dem Text codiert werden. Holistische Einschätzungen, inwiefern ein Text eine bestimmte Interpretation eines Sachverhalts nahelegt, sind automatisch nur schwer zu erkennen.

Beispiele für eine kleinteilige Erhebung inhaltlicher Kategorien lieferten Matthes und Kohring (2008) mit der analytischen Erfassung inhaltlicher Frames, sowie Semetko und Valkenburg (2000) mit der Messung generischer Frames über Skalen mit mehreren dichotomen Items. Für beide Vorgehensweisen müssen die Codierer konkrete und differenzierte Fragen zum Inhalt des Textes oder einzelner Aussagen beantworten, um Daten zu generieren, die später aufbereitet und zu inhaltlichen Kategorien verdichtet werden. Die Codierer nehmen also keine umfassenden Interpretationen vor, sondern codieren ausschliesslich das Vorhandensein einzelner, leicht erkennbarer Komponenten. Dies wirkt sich sowohl positiv auf die Reliabilität der Codierungen als auch auf die Effizienz der Codiererschulung aus (vgl. David, Atun, & La Viña, 2010). Gleichzeitig existiert damit ein Messinstrument, mit dem sich leichte Veränderungen in den Argumentationslinien oder Interpretationen messen und beschreiben lassen. Dies macht diese Art der Erhebung auch für longitudinale und komparative Studien interessant (vgl. Kohring & Matthes, 2002; Wirth et al. 2013; Lee & Maslog, 2005).

Während die Effizienz der Textakquise durch digitale Texte deutlich erhöht wird, bedeutet eine computerunterstützte Inhaltsanalyse bei der Entwicklung des Erhebungsinstruments mitunter einen Mehraufwand. So müssen hier trainierte Verfahren getestet, Verfahren für eine parallele automatische Textanalyse entwickelt und das Codebuch im Hinblick auf eine Einbindung in eine GUI formatiert werden. Inwiefern dieser Aufwand für Inhaltsanalyseprojekte gerechtfertigt ist, muss

aufgrund des Umfangs der Inhaltsanalyse, der Verfügbarkeit von Mitarbeitern in der Entwicklungsphase und möglicher Einsparungen durch Computerunterstützung in der Anwendungsphase beurteilt werden.

Kapitel 6. Computerunterstützte Testphase

Die Codiererschulung stellt die zentrale Aufgabe der Testphase dar und kann kaum durch den Einsatz von Computern optimiert werden. Vielmehr geht es in diesem Arbeitsschritt darum, das Codebuch und die Aufgaben, welche vor den Codierern stehen, so gründlich wie möglich zu vermitteln, um eine verlässliche und valide Codierung sicherzustellen. Der zweite Arbeitsschritt in dieser Phase ist der Eignungstest der Codierer, der später in der Anwendungsphase in Form eines laufenden Reliabilitäts- und Validitätstests wiederholt wird. Bei diesem Arbeitsschritt sind automatische Verfahren zur Berechnung und Analyse der Übereinstimmungen eine unverzichtbare Hilfe, da sie eine schnelle und exakte Berechnung unterschiedlicher Reliabilitätskoeffizienten liefern (vgl. Lombard et al., 2002).

6.1. Eignungs-, Reliabilitäts- und Validitätstests

Die Erfassung der Inhalte von Texten im Prozess einer Inhaltsanalyse ist ein Messverfahren. Wie bei anderen Messungen ist auch hier die Prüfung der Reliabilität und der Validität des Instruments zentral, wenn die Daten anschliessend verwendet werden sollen, um den untersuchten Gegenstand zu beschreiben (vgl. Lombard et al., 2002). Krippendorff (2013) unterscheidet bei Inhaltsanalysen zwischen der Reliabilität der Codierungen, die dann angenommen kann, wenn die Codierer unabhängig voneinander identische Analyseeinheiten gleich codieren und der Validität, die dann angenommen werden kann, wenn die Daten den Gegenstand unverzerrt abbilden. Eine gute Reliabilität führt dabei nicht zwingend zu einer guten Validität, da die Codierer sich auch in einer unzutreffenden Einschätzung einig sein können, eine schlechte Reliabilität steht jedoch einer validen Erhebung im Weg (vgl. Krippendorff, 2013: 269).

Für einen Test der Reliabilität des Erhebungsinstruments wird in der Regel eine kleine Anzahl von Texten von mehreren geschulten Codierern erfasst, um die Übereinstimmung der Codierungen zu prüfen. Stimmen die Codierer bei allen Kategorien in ihren Entscheidungen überein, so kann von einer reliablen Messung ausgegangen werden (vgl. Krippendorff 2013: 271; Kolb, 2004: 337). Die Validität kann anschliessend über die Übereinstimmung der einzelnen Codierentscheidungen mit einer sorgfältig durch die Projektleitung ausgearbeiteten Musterlösung geprüft werden (vgl. Krippendorff, 2013: 217; Kolb, 2004: 338). Nur wenn die Codierer untereinander und mit einer Musterlösung hinreichend übereinstimmen, sind die erhobenen Daten vertrauenswürdig und können für die Beantwortung von Forschungsfragen herangezogen werden (vgl. Lombard et al., 2002: 588; Popping, 2010: 1068; Krippendorff, 2013: 267f). Mangelnde Übereinstimmungen müssen über erneute Schulungen und gegebenenfalls präziseren Definitionen im Codebuch korrigiert werden (vgl. Rössler, 2010: 198). Selbst bei hinreichend grosser Reliabilität sind die Abweichungen kritisch zu

analysieren, da sie zu systematischen Verzerrungen der Daten führen können (vgl. Scharkow & Vogelgesang, 2015).

Trotz der vielbetonten Bedeutung von Reliabilitätstests zeigen systematische Reviews von Fachpublikationen mit Inhaltsanalyse-Daten, dass eine Dokumentation der Ergebnisse dieser Tests oftmals fehlen. In einer Studie von Lombard et al. (2002) über 200 Zeitschriftenartikel, welche zwischen 1994 und 1998 erschienen sind, enthielten nur 69% Angaben zur Reliabilität (S. 596). Zu ähnlichen Ergebnissen kamen auch Früh und Früh (2015), welche in nur zwei Dritteln der Artikel überhaupt Angaben zur Reliabilität fanden. In den meisten Fällen wurde lediglich ein Gesamtergebnis eines einzelnen Koeffizienten angegeben (S. 43). In der für diese Rahmenschrift durchgeführten Analyse enthielten nur 102 der 160 Artikel (64%) mit einer eigenen inhaltsanalytischen Erhebung Angaben zur Reliabilität. Nur die Hälfte davon (48 Artikel) berichteten dabei ein Zufallsbereinigtes Mass für Reliabilität. Die restlichen Artikel gaben lediglich die prozentuale Übereinstimmung der Codierer an. Die Quote der Dokumentation der Reliabilität bleibt also über die Zeit bei ca. zwei Dritteln der Artikel konstant.

Lombard et al. (2002) sehen zwei grundlegende Probleme, die für diesen Missstand in der Kommunikationswissenschaft verantwortlich sein könnten. Zum einen, so schliessen sie, besteht Unsicherheit über die korrekte Durchführung und Dokumentation von Reliabilitätstests (S. 588). An der Lösung dieses Problems beteiligen sich jedoch die Autoren aller Einführungsbücher, welche dem Reliabilitätstest jeweils eigene, leicht verständliche Kapitel widmen (z.B: Früh, 2009: 188ff; Rössler, 2010: 195ff; Krippendorff, 2013: 267ff; Mayring, 2007: 109ff). Zum anderen scheint die komplexe Berechnung einzelner Koeffizienten und der Mangel an technischen Hilfsmitteln einer Durchführung und Auswertung von Reliabilitätstests im Weg zu stehen (vgl. Lombard et al. 2002: 588). Da die Berechnung der einzelnen Reliabilitätskoeffizienten von Hand zu aufwändig und kompliziert ist, bietet sich der Einbezug geeigneter Computerprogramme an. Die Autoren leisten hier selber seit 2003 mit einer laufend aktualisierten Website (siehe: Lombard, Snyder-Duch, & Bracken, 2010) einen Beitrag, um geeignete Programme bekannt zu machen und damit diese die Hemmschwelle für die Durchführung von Reliabilitätstests zu senken.

Aus anekdotischer Erfahrung, die ich durch die Mitarbeit und Beratung inhaltsanalytischer Studien gesammelt habe, sehe ich eine weitere Hemmschwelle für die Dokumentation von Ergebnissen der Eignungs- und Reliabilitätstests. Die unterschiedlichen Koeffizienten, die gebräuchlich und in Lehrbüchern empfohlen sind, reagieren unterschiedlich stark auf bestimmte Fehlerquellen in den Daten und führen so häufig zu einer systematischen Über- oder Unterschätzung der Codierer, die von der Art der erhobenen Kategorien und der Grösse und Heterogenität des Codierteams abhängt. Flurgespräche über das Verhalten und die Angemessenheit von Koeffizienten für bestimmte

Anwendungen verunsichern und hemmen Forscher, die Ergebnisse ihrer Reliabilitätstests offen zu legen. Sei dies, weil die berechneten Koeffizienten gemeinhin als zu liberal betrachtet und nicht ernst genommen werden oder weil die Werte unterhalb der konventionell akzeptablen Grenzen liegen.

Um auch diese – leider kaum öffentlich oder wissenschaftlich diskutierte – Hemmschwelle für die Dokumentation von Reliabilitätstests abzubauen, möchte ich in dieser Dissertation einen zweiteiligen Beitrag leisten. Zum einen gehe ich in diesem Kapitel systematisch und anhand von Simulationsstudien der Frage nach, inwiefern bestimmte Randbedingungen der Erhebung die Reliabilitätskoeffizienten beeinflussen und zu einer Über- oder Unterschätzung der Reliabilität und der Leistung der Codierer führen können. Zum anderen habe ich in *Nogrod* eine Funktion für die Berechnung der gängigsten Reliabilitätskoeffizienten implementiert, mit der sich Tests über eine ausgewählte Anzahl Analyseeinheiten, Codierer und Variablen berechnen lassen. Eine codierer- und variablenspezifische Dokumentation zeigt auf, an welchen Stellen im Codierteam oder im Codebuch Probleme bestehen könnten. Mit diesem Hilfsmittel ist es möglich, auch in umfangreichen Inhaltsanalysen mit komplexen Codebüchern und grossen Codierteams eine differenzierte Analyse der Reliabilität und Validität der einzelnen Codierungen durchzuführen und in geeigneter Form zu präsentieren.

Hintergrund

Für die Dokumentation der Reliabilität und Validität von Inhaltsanalysen wurden in der Vergangenheit zig unterschiedliche Berechnungsarten und Koeffizienten vorgeschlagen (vgl. Krippendorff, 2013: 278). In der Forschung hat man sich jedoch weitestgehend auf die Verwendung einer Handvoll dieser Koeffizienten geeinigt (vgl. Lombard et al. 2002: 590). Zu all diesen Koeffizienten bestehen Annahmen und anekdotische Berichte, wie sie sich in der konkreten Anwendung auf Inhaltsanalysedaten verhalten, wann sie verzerrte Resultate liefern und in welchen Fällen sie dazu beitragen, die Reliabilität der Messung zu über- oder unterschätzen (vgl. Kolb, 2004; Lombard et al. 2002: 591). In diesem Abschnitt soll möglichst systematisch der Frage nachgegangen werden, welche dieser Annahmen haltbar sind und wie die gebräuchlichsten Koeffizienten (*Holsti*, *Alpha*, *Kappa* und *Pi*) auf unterschiedliche Fehlerarten und auf die Randbedingungen der Inhaltsanalysen reagieren. Zusätzlich wird auch der neue Koeffizient Lotus (vgl. Fretwurst, 2015) in die Analyse einbezogen, der entwickelt wurde, um Verzerrungen der paarweise berechneten Koeffizienten auszugleichen.

Reliabilitätskoeffizienten

Der am häufigsten verwendete Koeffizient zur Berechnung der Intercoderreliabilität ist die *prozentuale Übereinstimmung* der Codierungen oder der *Holsti-Koeffizient* (vgl. Hayes & Krippendorff, 2007: 80). Dieser Koeffizient gibt an, wie gross der Anteil an paarweise

übereinstimmenden Codierungen der einzelnen Codierer im Reliabilitätstest ist und ist damit ein sehr direktes und leicht verständliches Mass für die Übereinstimmung. Um diesen Koeffizienten zu berechnen wird für jedes Codiererpaar die Anzahl Übereinstimmungen durch die Anzahl codierter Einheiten geteilt. Der Durchschnitt der Ergebnisse, oftmals über alle Kategorien gemittelt, wird als Reliabilitätskoeffizient für die Erhebung gewertet (vgl. Lombard et al., 2002: 591).

Dieser Koeffizient wurde jedoch von verschiedenen Seiten als zu liberal eingestuft, da er auch zufällige Übereinstimmungen zwischen Codierern als Übereinstimmung wertet. Besonders bei schief verteilten Kategorien, bei denen eine zufällige Übereinstimmung besonders wahrscheinlich ist, resultieren sehr hohe Reliabilitätswerte, auch wenn die Codierer in vielen Fällen nicht übereinstimmen. Als Rechenbeispiel kann eine dichotome Variable angenommen werden, die in 90% der Fälle 0 und in 10% als 1 codiert wird. Bei dieser Variable ist die Wahrscheinlichkeit, dass beide Codierer zufällig gleichzeitig eine 0 codieren 81% und die Wahrscheinlichkeit, dass sie zufällig beide eine 1 codieren 1%. Selbst wenn diese Codierer also nach dem Zufallsprinzip arbeiten und die Texte nicht lesen, haben sie noch immer eine Übereinstimmung von 82%. Dazu ist jedoch auch zu sagen, dass sie auch mit der Musterlösung in 82% der Fälle übereinstimmen und die Erhebung damit in den meisten Fällen valide ist.

Um für zufällige Übereinstimmungen zu kontrollieren, gibt es die Möglichkeit, die prozentuale Übereinstimmung (*PA*) zu der erwarteten zufälligen Übereinstimmung bei dieser Kategorie (*PC*) in Beziehung zu setzen. Mit der allgemeinen Formel (PA-PC)/(1-PC) (vgl. Cohen, 1960) können Reliabilitätswerte berechnet werden, die maximal 1 sind, falls die prozentuale Übereinstimmung bei 100% liegt. Stimmen die Codierer nicht überzufällig überein, sondern erreichen genau die Übereinstimmung, die bei zufälliger Codierung erwartet würde, so ist der Koeffizient genau 0. Stimmen sie schlechter überein, als dies bei einer zufälligen Codierung zu erwarten wäre, wird der Koeffizient negativ. Der Wertebereich dieser Koeffizienten ist nach unten unbegrenzt, was in Einzelfällen zu stark negativen Werten führen kann.

Für die Berechnung der erwarteten zufälligen Übereinstimmung gibt es unterschiedliche Herangehensweisen, die in leicht verschiedenen Koeffizienten resultieren. Für die Berechnung von *Cohen's Kappa* (vgl. Cohen, 1960) wird für jedes Codiererpaar die erwartete zufällige Übereinstimmung anhand der Verteilung der Codierungen dieser beiden Codierer berechnet. Für die Berechnung von *Scott's Pi* (vgl. Scott, 1955) wird die erwartete zufällige Übereinstimmung über alle Codierer berechnet, um für jede Kategorie einen eindeutigen Wert zu erhalten. Für die Berechnung der *Alpha*-Koeffizienten von Krippendorff (2007; 2008) wird die durchschnittliche Übereinstimmung

verwendet, welche dann gefunden wird, wenn alle Codierungen aller Analyseeinheiten miteinander verglichen werden.

Der Vorteil dieser Koeffizienten liegt in der Bestimmung der überzufälligen Übereinstimmung der Codierer und damit in dem Mass, in welchem die Übereinstimmung tatsächlich durch den gemessenen Gegenstand erklärt werden kann. Der Nachteil dieser Koeffizienten liegt jedoch in ihrer Tendenz, bei schief verteilten Kategorien einzelne Codierfehler zu stark zu gewichten, was zu unannehmbar tiefen Werten führen kann. Als Rechenbeispiel kann hierfür ein Reliabilitätstest mit fünf Kategorien und 10 Texten angenommen werden, der von 10 Codierern mit beinahe perfekter Übereinstimmung codiert wird. Bei vier Kategorien machen sie keinen Fehler und stimmen zu 100% überein. Die fünfte Kategorie ist eine dichotome Entscheidung, ob ein Merkmal im Text auftritt. Dieses Merkmal kommt im Reliabilitätstest nicht vor und wird jeweils mit 0 codiert. Nur in einem Fall irrt sich ein Codierer und codiert fälschlicherweise eine 1. Die Variable wurde damit in 99 Fällen mit 0 codiert und in einem Fall mit 1, was eine erwartete zufällige Übereinstimmung nach Scott von $0.99^2+0.01^2=0.9802$ ergibt. Vergleicht man die Codierung des einen Codierers mit dem Fehler paarweise mit den anderen, so erhält man als Wert für den Koeffizienten *Pi* in allen Fällen (0.9-0.982)/(1-0.982) = -4.05.

Zwar wird in diesem Beispiel nur in neun der insgesamt 45 paarweisen Vergleiche der tiefe Wert von -4.05 erreicht, der Wert für den Koeffizienten *Pi* dieser Kategorie ist damit aber -0.1, was fälschlicherweise auf eine sehr schlechte Codierung hinweist. Auch im Durchschnitt über die fünf in beinahe perfekter Übereinstimmung codierten Kategorien ist *Pi* nur 0.8. Ab diesem Wert gilt eine Erhebung gemeinhin zwar noch als akzeptabel, aber nicht mehr als hervorragend (vgl. Lombard et al. 2002: 593). Schlimmer wird die Situation, wenn ähnliche Fehler auch in anderen Kategorien auftreten, und die zufallsbereinigten Koeffizienten Werte unter 0 annehmen.

Zusätzlich zu der hohen Sensitivität der zufallsbereinigten Koeffizienten vermutete Kolb (2004) bei paarweise berechneten Koeffizienten einen Einfluss der Anzahl der Codierer auf die realistisch erreichbaren Werte. Er schlug daher vor, die Übereinstimmung bei grossen Codierteams nicht paarweise, sondern über das Team als Ganzes zu berechnen (vgl. Kolb, 2004: 348). Dieser Vorschlag wurde von Fretwurst (2015) aufgegriffen, um den Koeffizienten *Lotus* zu entwickeln, bei dem jeweils der Anteil der Codierer berechnet wird, die mit der Mehrheit in einer Entscheidung übereinstimmen. Da dieser Koeffizient sehr liberal ist und vor allem bei Kategorien mit wenigen Ausprägungen bessere Werte erzielt als Holsti, kann zudem ein standardisierter Lotus berechnet werden, bei dem die Überschätzung durch wenige Kategorien korrigiert wird (S: 189).

Aus den Beschreibungen der Koeffizienten lässt sich schliessen, dass nicht nur die Qualität der Codierer, sondern auch andere Eigenschaften der Reliabilitätstests einen Einfluss auf die Grösse der Koeffizienten haben. So ist zu erwarten, dass die prozentuale Übereinstimmung besser und die zufallsbereinigten Koeffizienten schlechter werden, wenn die Ausprägungen einer Kategorie schief verteilt sind. Gleichzeitig fallen bei einer höheren Anzahl Codierer einzelne Abweichungen bei der Berechnung paarweiser Übereinstimmung weniger stark ins Gewicht, wodurch die paarweise berechneten Koeffizienten überschätzt werden könnten (vgl. Kolb, 2004: 342). Diese Annahmen und das konkrete Verhalten der Koeffizienten wurden einerseits mit einer Simulationsstudie und andererseits an einem realen Reliabilitätstest überprüft.

Simulationsstudie

Für die Simulationsstudie wurde ein fiktives Codierteam von 10 Codierern angelegt, die eine bestimmte Trefferquote bei der Codierung eines Reliabilitätstests haben und unsystematische Fehler begehen. Für den Reliabilitätstest wurde eine Musterlösung erstellt, welche bei der Simulation als Richtlinie für die korrekte Codierung dient. Der Test enthielt mehrere Kategorien mit unterschiedlich schiefer Verteilung. Für jede Kategorie wurden jeweils 30 Reliabilitätstests simuliert, um einen durchschnittlichen Reliabilitätskoeffizienten für die unterschiedlich schief verteilten Kategorien zu berechnen. Diese Simulation wurde für unterschiedliche Trefferquoten der Codierer und unterschiedlich grosse Teams simuliert, um den Einfluss der Codierqualität und der Anzahl Codierer auf die Ergebnisse bei unterschiedlich schief verteilten Kategorien zu testen.

Mit der Anzahl der Codierer, der Trefferquote bei der Codierung, der Verteilung der Kategorien und den unterschiedlichen berechneten Koeffizienten wurden vier Faktoren variiert, was eine gleichzeitige Darstellung und Beschreibung aller Ergebnisse in einer Grafik oder Tabelle verunmöglicht. Für einen Überblick wird darum die Standardsituation angenommen, dass es sich um ein 10-köpfiges gutes Codierteam mit 90% Übereinstimmung mit Musterlösung handelt, welches eine dichotome Variable mit einer moderat schiefen Verteilung (70:30) codiert. Für den Reliabilitätstest wurden alle Koeffizienten gleichzeitig berechnet. Durch die Variation jeweils eines einzelnen Faktors kann dessen Einfluss auf die Ergebnisse geprüft werden.

Um die Abhängigkeit der Resultate von der Anzahl Codierer zu bestimmen, wurden die Tests für unterschiedlich grosse Teams zwischen 2 und 10 Codierern ausgewertet. Das Resultat (Abbildung 11) zeigt deutlich, dass die Befürchtung von Kolb (2004) unbegründet ist und die Anzahl Codierer keinen Einfluss auf paarweise berechnete Koeffizienten hat, sofern alle Codierer die gleiche Fehlerquote haben und zufällige Fehler begehen. Bei gleicher Qualität und gleicher Zusammensetzung der Kategorien führen alle Tests auch für unterschiedlich grosse Teams zum selben Resultat.

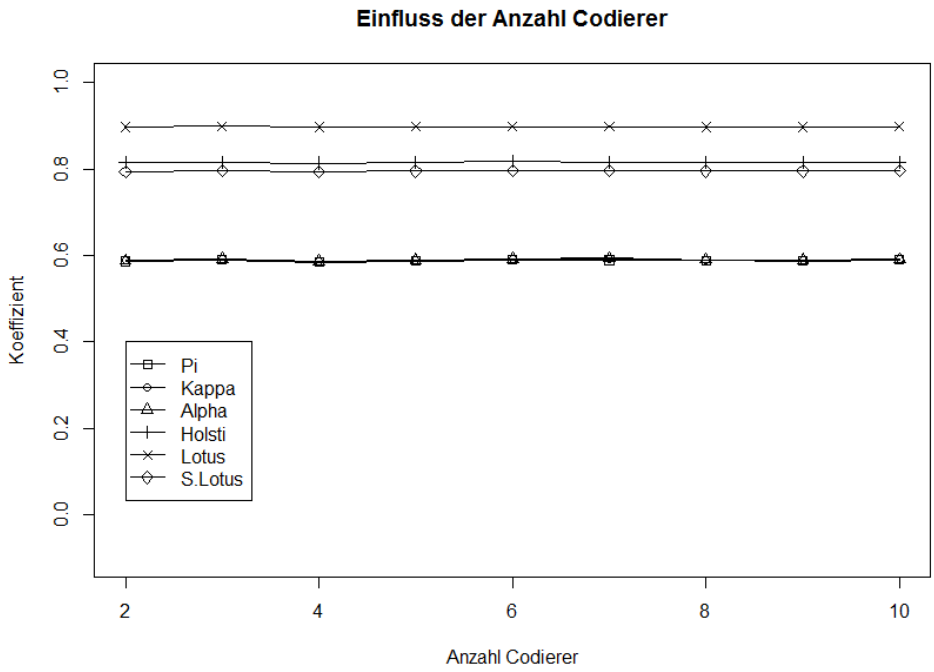


Abbildung 11: Einfluss der Anzahl Codierer auf die Reliabilitätskoeffizienten. Für die Simulation wurden 2-10 Codierer mit einer Zuverlässigkeit von 90% bei der Codierung einer dichotomen Variable mit einer moderat schiefen Verteilung (70:30) verwendet. Die Linien für Pi, Kappa und Alpha liegen beinahe exakt übereinander auf Höhe 0.6.

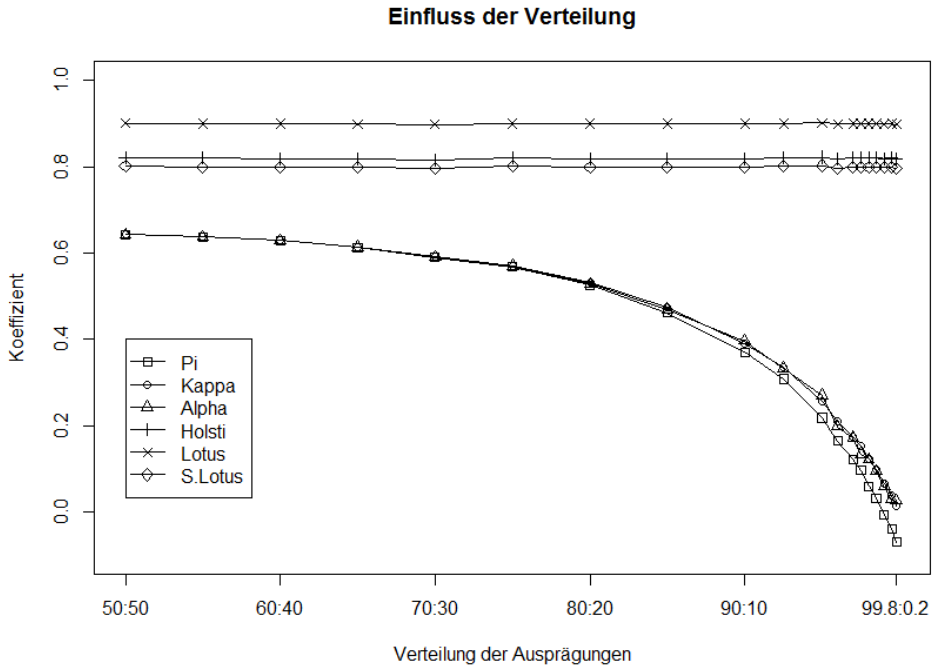


Abbildung 12: Einfluss der Schiefe der Verteilung der Ausprägungen auf die Reliabilitätskoeffizienten. Für die Simulation wurden 10 Codierer mit einer Zuverlässigkeit von 90% bei der Codierung einer dichotomen Variable mit unterschiedlich schiefen Verteilungen verwendet.

Um in einer weiteren Simulation die Abhängigkeit der Resultate von der Schiefe der Verteilung der Ausprägungen zu testen, wurden Dichotome Kategorien mit Verteilungen zwischen 50:50 (gleichmässige Verteilung) bis 99.8:0.2 (sehr schiefe Verteilung) simuliert. Das Resultat in Abbildung 12 zeigt, dass die zufallsbereinigten Koeffizienten (*Alpha*, *Kappa*, *Pi*) bei schiefer verteilten Kategorien und gleicher Trefferquote kleiner werden. Sie nähern sich trotz einer 90% korrekten Codierung dem Wert 0 an. Auf die prozentuale Übereinstimmung, sowie die Lotus-Koeffizienten hat eine schiefe Verteilung hingegen keinen Einfluss.

Schliesslich wurde die Abhängigkeit der Resultate von der Codierqualität getestet. Hier wurde die Trefferquote der Codierer von 50% bis 100% (perfekte Codierung) variiert. Das Ergebnis in Abbildung 13 zeigt, dass alle Koeffizienten von der Qualität der Codierungen abhängen und bei einer perfekten Codierung den Wert 1 annehmen. Gleichzeitig kann jedoch an dieser Darstellung auch abgelesen werden, dass die Koeffizienten bei nur 50% Trefferquote der Codierer unterschiedlich hoch sind. In der Realität würde eine so tiefe Trefferquote bedeuten, dass die Codierer jeweils per Münzwurf entschieden haben, welche Ausprägung sie codieren möchten. Die Reliabilität einer solchen Messung ist also als sehr schlecht einzustufen. Dennoch beträgt Lotus in diesem Fall 0.64. Ein Wert, welchen die zufallsbereinigten Koeffizienten erst ab einer Übereinstimmung von über 90% annehmen können.

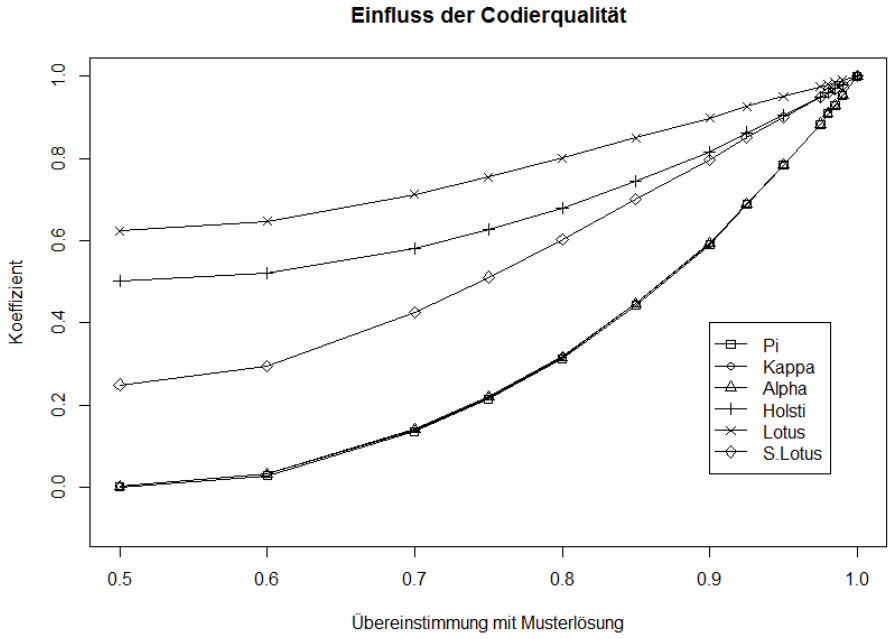


Abbildung 13: Einfluss der Qualität der Codierer auf die Reliabilitätskoeffizienten. Für die Simulation wurden 10 Codierer mit einer Zuverlässigkeit von 50-90% bei der Codierung einer dichotomen Variable mit einer moderat schiefen Verteilung (70:30) verwendet. Die zufallsbereinigten Koeffizienten (Pi, Kappa, Alpha) unterscheiden sich nicht.

Bei der Dokumentation eines Reliabilitätstests mit Lotus oder Holsti ist der Wertebereich der Koeffizienten in die Interpretation einzubeziehen. Liegt das Ergebnis für die gesamte Analyse bei einem Holsti von 0.5 oder Lotus von 0.6, so kann nicht von einer verlässlichen Messung ausgegangen werden. Gleichzeitig kann ein zufallsbereinigter Koeffizient von 0.3 bereits auf eine Übereinstimmung der Codierer von über 80% hinweisen und für eine Analyse ausreichend sein.

Bei allen drei Simulationen zeigte sich, dass die drei zufallsbereinigten Koeffizienten (*Cohen's Kappa, Scott's Pi und Krippendorff Alpha*) immer beinahe dieselben Werte annahmen. Da alle Codierer exakt dieselbe Trefferquote haben und nur zufällige Fehler begehen, führen die unterschiedlichen Berechnungsarten der erwarteten zufälligen Übereinstimmung jeweils zu sehr ähnlichen Ergebnissen. Unterschiede zwischen diesen Koeffizienten sollten jedoch zu erwarten sein, wenn das Codierteam eine heterogene Leistung aufweist. Um diese Annahme zu testen, wurden zwei weitere Simulationen durchgeführt, bei welchen in einem Fall ein durchmischtes Codierteam mit Leistungen zwischen 50% und 95% den Test machte, und im anderen Fall ein gutes Codierteam (9 Codierer mit 90% Trefferquote) einen einzelnen schlechten Codierer mit einer Trefferquote von 50% enthielt.

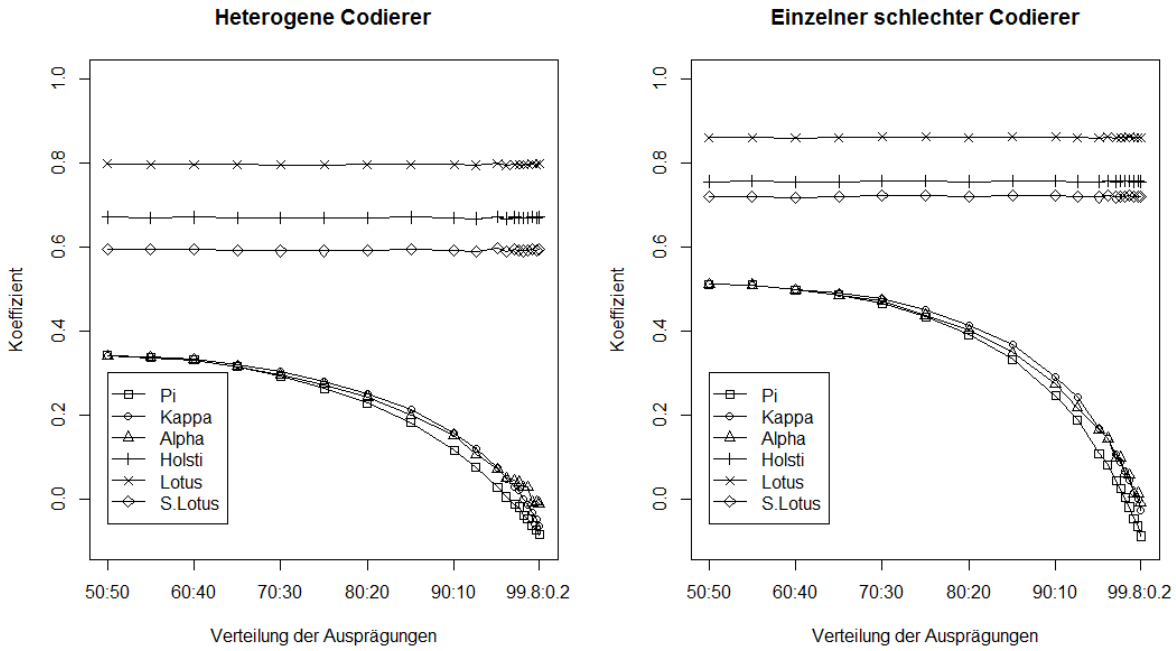


Abbildung 14: Einfluss heterogener Codierteams auf die einzelnen Koeffizienten. Für die Simulation wurde einmal mit einem gemischten Team aus 10 guten und schlechten Codierern (links), und einmal mit neun guten Codierern (90%) und einem schlechten (50%) gerechnet.

Das Ergebnis dieser Simulationen ist in Abbildung 14 dargestellt und zeigt deutlich, wie die Koeffizienten auf heterogene Leistungen im Codierteam reagieren. Insbesondere zeigt sich, dass Scott's Pi sensibler auf ungleiche Leistungen reagiert. Sei dies im Fall eines heterogenen Teams oder bei der schlechten Leistung eines einzelnen Codierers. Auch hier kann jedoch festgestellt werden, dass die zufallsbereinigten Koeffizienten jeweils nahe beieinander liegen und prinzipiell austauschbar sind.

Realdaten

In der Simulation wurde mit zufällig generierten Daten gearbeitet, deren Fehler sich unsystematisch über den gesamten Test verteilten. In realen Erhebungen ist jedoch mit systematischen Verzerrungen, vereinzelten zweideutigen Analyseeinheiten und unterschiedlich schwierigen Kategorien zu rechnen. Um das Verhalten der Koeffizienten unter realen Bedingungen zu testen, wurden alle Kategorien der Inhaltsanalyse des NCCR-Moduls " *Populismus in Zeiten der Globalisierung und Mediatisierung*", unabhängig von deren Skalenniveau und möglicherweise erforderlichen Aggregationen, in einen Reliabilitätstest einbezogen. Dabei wurden auch drei Codierer berücksichtigt, die aufgrund ihrer schlechten Leistung aus dem Codierteam ausgeschlossen wurden.

Insgesamt wurden im Reliabilitätstest 15 Texte von 28 Codierern bearbeitet. In diesen Texten wurden jeweils alle Sprecher und deren Aussagen codiert, womit der Reliabilitätstest insgesamt 347 Analyseeinheiten umfasste (15 Texte, 87 Sprecher, 114 Aussagen zu Themen, 131 Aussagen über Akteure). Über alle Analyseeinheiten wurden 219 Kategorien in den Reliabilitätstest einbezogen. Die Codierer stimmten untereinander mit durchschnittlich 88.9% überein, was einem *Scott's Pi* von 0.230 entsprach (*Kappa*=0.652; *Alpha*=0.241; *Lotus*=.931). Durch systematische Codierfehler einzelner Codierer bei sehr selten vorkommenden Ausprägungen nahm *Scott's Pi* für einzelne Kategorien Werte bis -2.5 an, was sich sehr negativ auf die Gesamtwertung auswirkte.

Auffällig am Gesamtergebnis ist die hohe Ähnlichkeit zwischen *Scott's Pi* und *Krippendorff Alpha*. Auch wenn in der Literatur darauf hingewiesen wird, dass diese Koeffizienten sich für hinreichend grosse Fallzahlen einander annähern (vgl. Krippendorff, 2013: 305f), ist die Ähnlichkeit frappierend. Insbesondere, da beide stark von *Cohen's Kappa* abweichen. Eine detailliertere Aufschlüsselung des Zusammenhangs zwischen *Pi* und *Alpha* zeigt, dass tatsächlich ein linearer Zusammenhang zwischen diesen beiden Koeffizienten besteht. Je höher *Scott's Pi* für eine Variable ist, desto höher ist auch *Krippendorff Alpha* (siehe Abbildung 15).

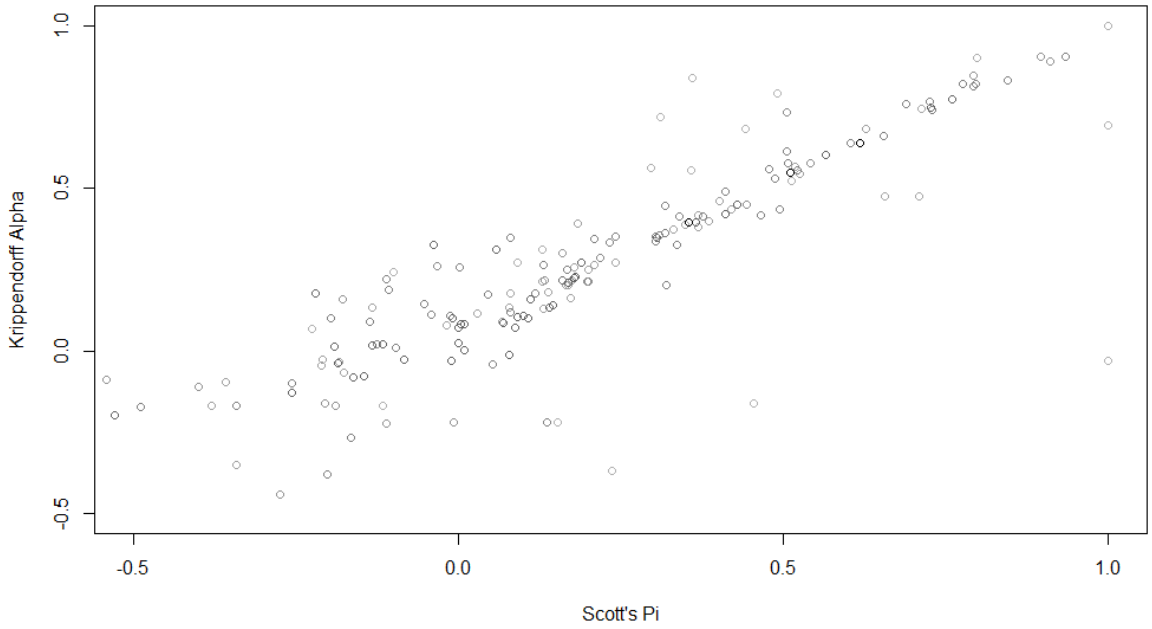


Abbildung 15: Zusammenhang zwischen Krippendorff Alpha und Scott's Pi unter realen Bedingungen. Die Kreise stehen für die 219 in den Reliabilitätstest einbezogenen Kategorien. Dunklere Kreise wurden im Reliabilitätstest häufiger codiert, hellere Kreise seltener.

Die hohe prozentuale Übereinstimmung in diesem Reliabilitätstest und die tiefen Werte für *Scott's Pi* und *Krippendorff Alpha* lassen vermuten, dass viele schief verteilte Kategorien im Test enthalten waren, bei denen die Kappa-Werte jeweils stark von der prozentualen Übereinstimmung abweichen. Das vergleichsweise gute Resultat von *Cohen's Kappa* lässt hingegen auf eine heterogene Leistung im Codierteam schliessen. Nicht alle Codierer haben also gleich gut gearbeitet.

Um den Einfluss schief verteilter Kategorien genauer zu betrachten, wurde für jede Variable die Entropie der Verteilung ihrer Ausprägungen im gesamten Reliabilitätstest berechnet. Eine Entropie von 0 bedeutet, dass eine der Ausprägungen zu 100% gewählt wurde. Eine Entropie von 1 bedeutet, dass alle Ausprägungen mit der gleichen Häufigkeit codiert wurden (vgl. Hellman, 2001). Der Zusammenhang zwischen *Scott's Pi* und der Verteilung ist in Abbildung 16 dargestellt. An dieser Darstellung lässt sich (links) ablesen, dass *Scott's Pi* für alle Kategorien tiefer war als die prozentuale Übereinstimmung. Besonders bei Kategorien mit hoher prozentualer Übereinstimmung ist *Pi* mitunter klein und kann auch negative Werte annehmen. Deutlicher wird dieser Zusammenhang im rechten Teil der Darstellung, die zeigt, dass *Pi* vor allem in jenen Fällen negativ wird, in welchen fast ausschliesslich dieselbe Ausprägung codiert wurde. In Fällen mit hoher Entropie ist *Pi* dagegen meist positiv. Dieser Koeffizient bestraft also Fehler in schief verteilten Kategorien deutlich stärker.

Ein ähnliches, wenn auch weniger ausgeprägtes Verhalten zeigt *Cohen's Kappa* bei schief verteilten Kategorien (siehe Abbildung 17). Hier kann jedoch zudem beobachtet werden, dass es bei sehr tiefer Entropie, also einer sehr schiefen Verteilung der Kategorie, ein Mindestwert für *Kappa* zu geben scheint, welcher nie unterschritten wird. Dies hängt damit zusammen, dass die zufällig erwartete Übereinstimmung bei diesem Koeffizienten für jedes Codiererpaar separat berechnet wird und damit bei Codiererpaaren, die nicht schlechter übereinstimmen, tiefer liegt als bei Paaren, die besser übereinstimmen.

Als Kontrast zum beobachteten Verhalten von zufallsbereinigten Koeffizienten kann an dieser Stelle auf das gegenläufige Verhalten von *Lotus* (siehe Abbildung 18) hingewiesen werden. Für diesen Koeffizienten erkennt man, dass er wie bereits in den Simulationsstudien fast immer über dem *Holsti*-Koeffizienten liegt. Zudem kann festgestellt werden, dass *Lotus* vor allem für Kategorien mit sehr schiefen Verteilungen sehr hohe Werte annimmt. Die hohen Werte sind in diesen Fällen scheinbar Unabhängig von der Qualität der einzelnen Codierer. Nur bei Kategorien mit gleichmässig verteilten Ausprägungen ist bei *Lotus* Varianz erkennbar.

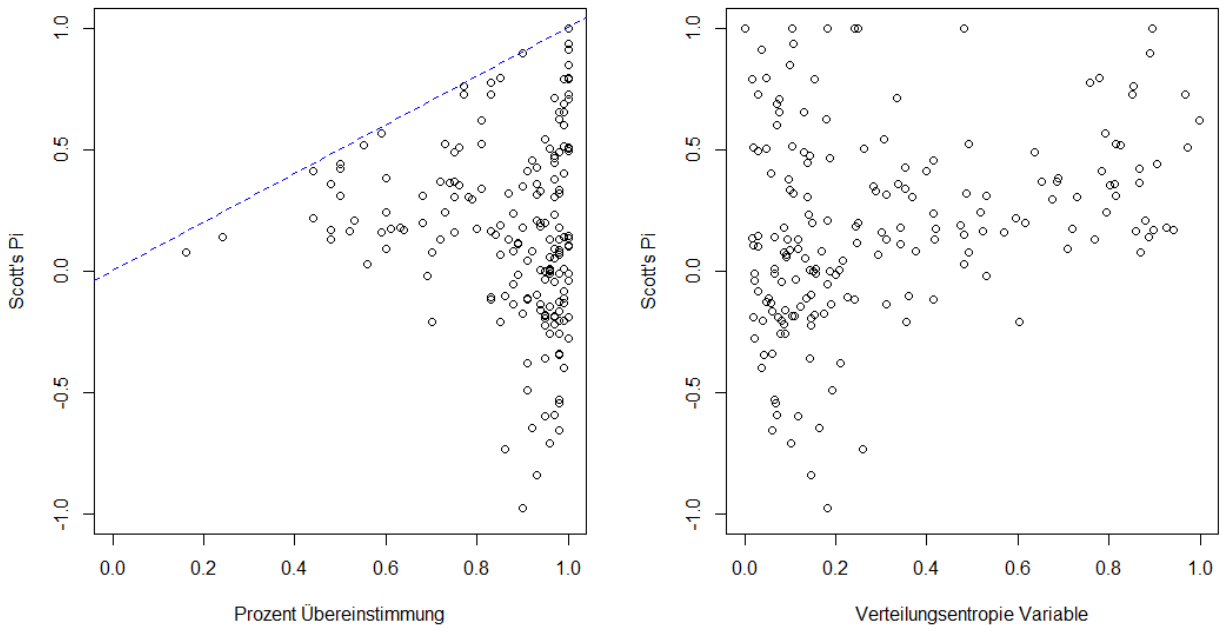


Abbildung 16: Zusammenhang zwischen der durchschnittlichen prozentualen Übereinstimmung und der Verteilungsentropie einzelner Kategorien auf und dem Reliabiltätskoeffizient Scott's Pi. Bei Kategorien unterhalb der gestrichelten Linie ist die prozentuale Übereinstimmung höher als Scott's Pi. Bei Kategorien über der gestrichelten Linie ist Scott's Pi höher.

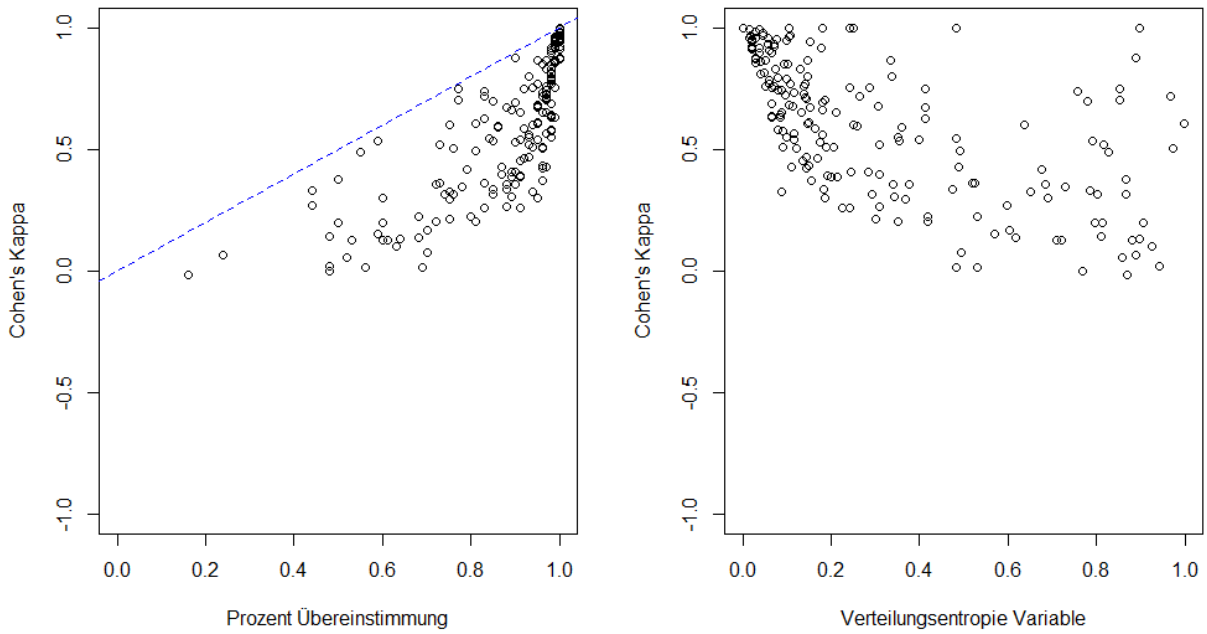


Abbildung 17: Zusammenhang zwischen der durchschnittlichen prozentualen Übereinstimmung und der Verteilungsentropie einzelner Kategorien auf und dem Reliabiltätskoeffizient Cohen's Kappa. Bei Kategorien unterhalb der gestrichelten Linie ist die prozentuale Übereinstimmung höher als Cohen's Kappa. Bei Kategorien über der gestrichelten Linie ist Kappa höher.

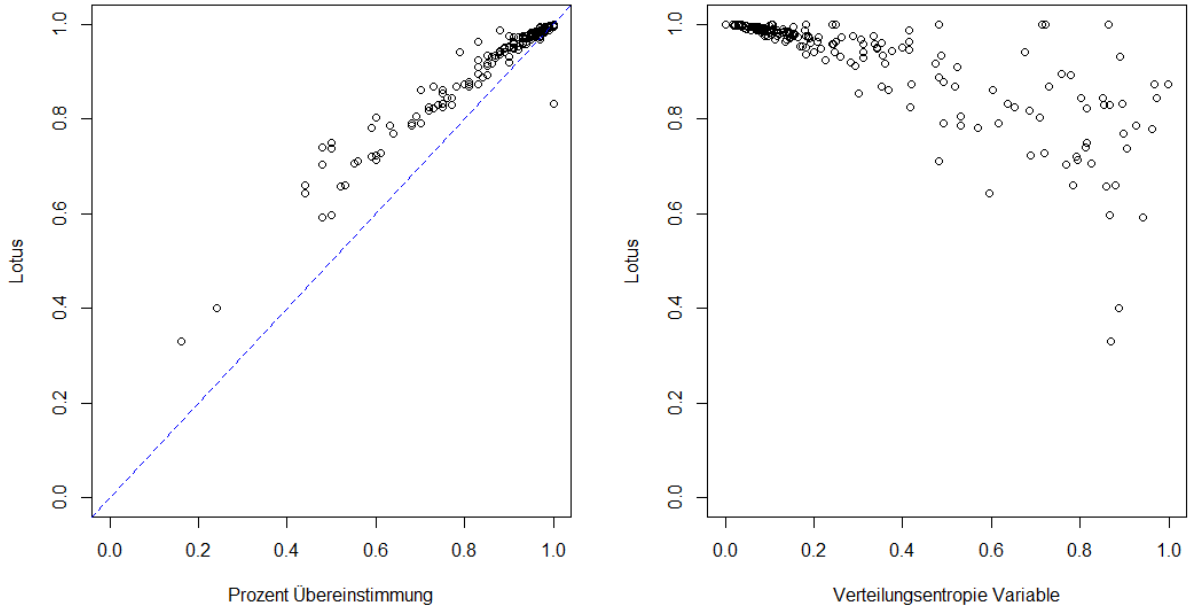


Abbildung 18: Zusammenhang zwischen der durchschnittlichen prozentualen Übereinstimmung und der Verteilungsentropie einzelner Kategorien auf und dem Reliabiltätskoeffizient Lotus. Bei Kategorien unterhalb der gestrichelten Linie ist die prozentuale Übereinstimmung höher als Lotus. Bei Kategorien über der gestrichelten Linie ist Lotus höher.

Diskussion und Empfehlungen

Sowohl in der Simulation als auch in der Realität lassen sich Besonderheiten der häufig eingesetzten Reliabilitätskoeffizienten festmachen. Erstens kann festgehalten werden, dass die Anzahl der Codierer in einem Reliabilitätstest das Ergebnis weder positiv noch negativ beeinflussen. Lediglich das Vorhandensein sehr schlechter Codierer kann einen schädlichen Einfluss auf einzelne Koeffizienten haben. Zweitens gilt es zu beachten, dass die zufallsbereinigten Koeffizienten besonders bei schief verteilten Kategorien sehr tiefe – unter Umständen stark negative – Werte annehmen und damit das Gesamtergebnis negativ beeinflussen können, da Fehlentscheidungen der Codierer bei diesen Kategorien überbewertet werden. Schliesslich zeigte sich auch, dass *Scott's Pi* und *Krippendorff Alpha* in den meisten Fällen austauschbar sind und bei heterogenen Codierteams tiefere Werte annehmen als *Cohen's Kappa*. *Lotus* überschätzt hingegen die Codierqualität massiv, wodurch hier bereits Werte unter 0.9 als problematisch angesehen werden müssen.

Für die Durchführung von Reliabilitätstests bietet es sich an, mehrere Koeffizienten zu berechnen und zu analysieren bei der Dokumentation kann auf einzelne Kategorien hingewiesen werden, deren Erhebung nicht als verlässlich gilt. Falls eine Kategorie durch ihre schiefe Verteilung im Reliabilitätstest zu schlechten Ergebnissen führt, kann dies offen diskutiert und analysiert werden. Zudem kann das Resultat der Reliabilitätstests belastbarer gemacht werden, wenn darauf geachtet wird, dass die wichtigen Kategorien der Inhaltsanalyse gleichmässig verteilt sind. Durch eine geschickte Auswahl der Texte für den Reliabilitätstest kann eine gleichmässige Verteilung auch für Kategorien erzwungen werden, die in der Realität sehr schief verteilt sind.

Im Minimum sollte ein Reliabilitätstest die prozentuale Übereinstimmung und mindestens einen zufallsbereinigten Koeffizienten enthalten. Falls sich die unterschiedlichen zufallsbereinigten Koeffizienten unterscheiden, können diese Unterschiede untersucht und in der Dokumentation erörtert werden.

Kapitel 7. Computerunterstützte Anwendungsphase

In der Anwendungsphase findet mit der Erhebung in den meisten Inhaltsanalysen der aufwändigste und langwierigste Arbeitsschritt statt. In diesem Schritt werden die Texte durch trainierte Codierer gelesen und mit Hilfe des Erhebungsinstruments erfasst. Bei dieser Aufgabe müssen die Codierer einerseits betreut und regelmässig mit neuem Material versorgt werden, andererseits muss die Qualität ihrer Arbeit, also die Validität ihrer Codierentscheidungen überprüft werden. Nach der Erhebung folgt mit der Datenaufbereitung der finale Schritt, an dessen Ende ein Datensatz zur Beantwortung der Forschungsfragen vorliegt.

Computerunterstützung ist in dieser finalen Phase der Inhaltsanalyse an mehreren Stellen möglich. So sind die Berechnung der Reliabilität und Validität der Codierer, der Vorgang der Dateneingabe, sowie die Betreuung und Versorgung der Codierer und die Datenauswertung durch elektronische Hilfsmittel möglich und ratsam, um die Qualität und Effizienz der Inhaltsanalyse zu steigen. Da der Reliabilitätstest in Kapitel 6.1 bereits erschöpfend diskutiert wurde, wird darauf in diesem Kapitel nicht mehr eingegangen.

7.1. Erhebung

Im Arbeitsschritt der Erhebung sind die Codierer damit betraut, das Erhebungsinstrument auf die ihnen zugewiesenen Texte aus dem Textkorpus anzuwenden und sie damit auf für das Forschungsinteresse relevante Informationen zu abstrahieren. In qualitativen Inhaltsanalysen geschieht dies durch Zusammenfassung, Kategorisierung und Interpretation der Texte anhand eines Leitfadens (vgl. Mayring, 2007), in quantitativen Inhaltsanalysen werden die Texte gegebenenfalls in Analyseeinheiten unterteilt, um darin Codiereinheiten zu finden und nach Vorgaben des Codebuches zu klassifizieren, zählen oder vermessen (vgl. Mayring, 2007: 14-15; Früh, 2009: 113).

Während diese Aufgabe bei manuellen, also auch bei computerunterstützten, Inhaltsanalysen per Definition hauptsächlich von Hand durchgeführt wird, bieten sich auch hier elektronische Hilfsmittel an, mit denen die Erhebung effizienter oder verlässlicher ausgeführt werden kann. Auf die Möglichkeit einer parallelen automatischen Erhebung für einzelne Kategorien wurde bereits in der Beschreibung der Entwicklungsphase eingegangen. Zusätzlich zu dieser Möglichkeit bietet sich auch durch die manuelle Eingabe der Daten in ein geeignetes Computerprogramm an. In diesem Kapitel sollen die Einsatzmöglichkeiten und die Vorzüge einer computerunterstützten Dateneingabe diskutiert werden.

7.1.1. Dateneingabe über eine Eingabemaske

Im aktuellsten deutschsprachigen Lehrbuch schreibt Rössler (2010), dass die Eingabe der Daten direkt am Computer in gewissen Fällen Vorzüge haben kann. Aber "klassischerweise werden die Codierresultate auf Papier fixiert und anschliessend in den Rechner eingegeben" (S. 184). Diese Praxis scheint insbesondere für quantitative Inhaltsanalysen noch heute verbreitet. In Fällen, in denen dennoch Computer für die Eingabe quantitativer Daten genutzt werden, geschieht dies meist in Tabellen in Excel oder SPSS, die den papiernen Codierbögen nachempfunden sind. Durch diesen Einsatz von Computern wird lediglich die Niederschrift auf einen Codierbogen und die Übertragung in eine Tabelle übersprungen und die Dateneingabe direkt dem Codierer überlassen.

In der qualitativen Inhaltsanalyse besteht hingegen eine längere Tradition, elektronische Eingabemasken für die Eingabe von Inhaltsanalysedaten zu nutzen. Anders als bei der quantitativen Analyse werden bei der qualitativen Analyse während der Erhebung neue Kategoriensysteme entwickelt und mit Textpassagen verknüpft. Manuell bedeutet diese Arbeit den Einsatz von Karteikarten sowie das Kopieren und Ausschneiden von Textfragmenten. Auf Papier sind diese Arbeitsschritte zeit- und kostenaufwändig, automatisch lassen sie sich jedoch bereits mit geringem Aufwand realisieren (vgl. Luzar, 2004: 147). Aus diesem Grund sind hier seit den 90er Jahren mit *Atlas/ti* und *MaxQDA* weit verbreitete Softwarepakete verfügbar, welche diese Arbeit am Bildschirm ermöglichen und erleichtern (vgl. Kuckartz, 1996; 2007; Muhr, 1996).

In der quantitativen Inhaltsanalyse sind erst wenige Anwendungen für eine Dateneingabe über Eingabemasken bekannt (vgl. Macnamara, 2005). Das wohl fortschrittlichste heute verfügbare System ist das in Amsterdam entwickelte *AmCAT* (vgl. van Atteveldt, Ruigrok, Takens, & Jacobi, 2014), mit dem sowohl manuelle als auch automatische quantitative Inhaltsanalysen über ein Onlineformular durchgeführt und mit Daten aus unterschiedlichen Projekten in Zusammenhang gebracht werden können. Weil *AmCAT* frei und als Online-Ressource ohne Installation verfügbar ist, kann es mit relativ geringem Aufwand für eine quantitative Inhaltsanalyse eingesetzt werden.

Ein ähnliches Programm, das im Rahmen dieser Dissertation spezifisch für die Erhebung anhand hierarchischer Codebücher entwickelt wurde, ist das Codierer-Interface *Angrist* (vgl. Wettstein, 2014a). Bei diesem Programm handelt es sich nicht um ein Online-Formular, sondern um ein ausführbares Python-Skript, das den Codierer durch die komplexe Aufgabe führt, einen Text in mehrere - möglicherweise verschachtelte - Analyseeinheiten zu unterteilen, um diese zu codieren.

Die Entwicklung dieses Programm stützte sich auf die von Wirth (2001) angestellten theoretischen Überlegungen zum Codierprozess als gelenkte Medienrezeption durch den Codierer. Anders als bei der alltäglichen Medienrezeption muss der Codierer sich bei der Rezeption eines Textes durch das Codebuch leiten lassen, um den Text systematisch zu erfassen (S. 163). Ausgehend

von dieser Überlegung wurde *Angrist* so konzipiert, dass es eine Reihe von Fragen an den Codierer stellt, die sich schnell durch die Auswahl einer Option in einer Liste oder die Markierung von Kästchen beantworten lassen.

Als ein grosses Problem für die Verlässlichkeit von Inhaltsanalysedaten stellt Wirth (2001) die heuristische Verarbeitung der Informationen im Text heraus. Verarbeitet der Codierer die Information nicht systematisch, sondern lässt sich von Gefühlen und einfachen Entscheidungsheuristiken leiten, so schadet dies der Reliabilität der Messung (S. 164ff). Eine solche heuristische Verarbeitung ist insbesondere dann zu beobachten, ein Rezipient durch eine zweite, mental anstrengende Aufgabe beansprucht wird (vgl. Kahneman, Beatty & Pollack, 1967; Petty, Wells & Brock, 1976). In Inhaltsanalysen kann die Codierarbeit selber eine derartige Zusatzaufgabe darstellen, wenn der Codierer häufig im Codebuch nachschlagen und sich an Codes, Beispiele und Anweisungen erinnern muss.

Um diese Fehlerquelle während der Erhebung zu minimieren stellt *Angrist* alle Fragen in natürlicher Sprache und bietet dem Codierer die Antwortoptionen nicht in Form von Codes, sondern über ihre Bezeichnung im Codebuch an. Gleichzeitig verfügt es über eine Hilfe-Funktion, mit der die Codebuchdefinition zu jeder Kategorie am Bildschirm nachgeschlagen werden kann. Der Codierer muss sich damit nur auf den Bildschirm konzentrieren und kann fokussiert arbeiten. Im Hintergrund werden die eingegebenen Daten in relationalen Datenbanken abgelegt, wobei die einzelnen Analyseeinheiten innerhalb eines Textes automatisch verknüpft werden. Die Gefahr, dass ein Codierer bei der Benennung und Verknüpfung von Analyseeinheiten einen Fehler begeht (vgl. Rössler 2010: 81) wird dadurch minimiert.

Eingabemasken oder GUIs (graphical user interfaces), welche die Eingabe von Daten am Bildschirm in dieser Form erlauben, haben für die Inhaltsanalyse zwei entscheidende Vorteile. Einerseits wird durch die Ausformulierung der Kategorien und Codes während der Dateneingabe die Gefahr minimiert, dass der Codierer Codes verwechselt oder Zahlen einträgt, welche nicht als Code definiert sind. Andererseits reduzieren Sie den kognitiven Aufwand des Codierers, indem sie Schrittweise durch die Erhebung führen und ihn nicht mit der Verwaltung von Tabellen und Identifikationsnummern belasten. Der Einsatz von GUIs zur Datenerhebung ist damit auch für die quantitative Inhaltsanalyse ratsam und durch die frei verfügbaren Programme, die in den letzten Jahren im akademischen Umfeld entwickelt wurden und noch immer entwickelt werden, deutlich erleichtert worden.

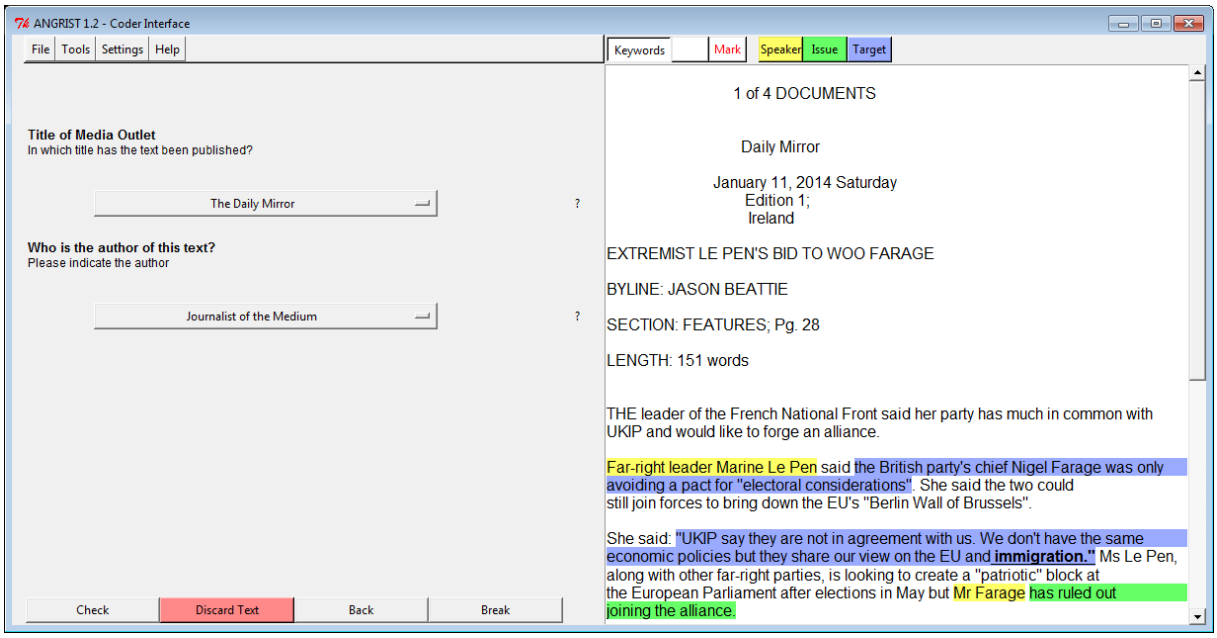


Abbildung 19: Ansicht eines Textes in Angrist. Rechts wird der Text dargestellt. Analyseeinheiten können mit unterschiedlichen Farben markiert werden. Rechts werden konkrete Fragen zum Text in natürlicher Sprache gestellt. Der Codierer kann über die Auswahl aus Menüs oder Listen oder durch andere Eingabehilfen die Daten eingeben. Sind alle Fragen beantwortet kann über den Knopf 'Check' die nächste Seite des Fragebogens aufgerufen werden. Hilfestellungen, benutzerdefinierte Einstellungen der Oberfläche und das Speichern und Laden von Codierungen erleichtern dem Codierer die Arbeit.

7.1.2. Halbautomatische Erfassung von Texten

Wenn die Daten einer quantitativen Inhaltsanalyse über eine Eingabemaske eingegeben werden, bietet sich die Möglichkeit, eine halbautomatische Inhaltsanalyse zu realisieren (vgl. Wettstein, 2014b). Bei dieser Art der Erhebung stellt die Eingabemaske die Ausprägungen der einzelnen Kategorien nicht nur dar, sondern verwendet automatische Textanalyseverfahren um den Codierer bei der Auswahl zu unterstützen. Während der Codierer alle finalen Codierentscheidungen trifft, helfen ihm Vorschläge des Programms dabei, schnell die zutreffenden Optionen zu finden.

In meinem Beitrag (2014b) skizziere ich unterschiedliche Möglichkeiten, mittels automatischer Verfahren Voraussagen zu treffen, die den Codierer bei seiner Arbeit unterstützen. Einerseits lassen sich regelbasierte Algorithmen nutzen, um den Text nach konkreten Informationen wie dem Erstelldatum, dem Autor oder der Anzahl Worte zu durchsuchen. Diese Informationen können dem Codierer zur Bestätigung angeboten werden, wodurch er nur eine Kontrollfunktion übernimmt.

Dieselben Verfahren können verwendet werden, um die Namen von Akteuren zu identifizieren, die im Text oder im aktuell bearbeiteten Absatz genannt werden. Mit dem Ergebnis dieser Analyse können Listen von Akteuren auf den Text zugeschnitten und um nicht vorkommende Akteure reduziert werden. Dies erleichtert die Suche nach den korrekten Akteuren und steigert die Effizienz der Codierer.

Zusätzlich können für die halbautomatische Inhaltsanalyse auch trainierte Verfahren eingesetzt werden, die anhand bereits codierter Texte die Wahrscheinlichkeit berechnen, dass der Codierer eine bestimmte Ausprägung auswählen wird. Die wahrscheinlichsten Ausprägungen können dem Codierer als Vorschläge angeboten werden. In Fällen, in denen mit hoher Wahrscheinlichkeit von einer bestimmten Ausprägung ausgegangen wird, kann diese durch die Eingabemaske automatisch angewählt werden. Der Codierer muss die Auswahl nur noch bestätigen oder korrigieren. In ambigen Situationen kann dem Codierer eine Auswahl der plausibelsten Optionen angeboten werden, aus denen er die zutreffende aussuchen kann. Falls keine klare Prognose möglich ist, sollte hingegen auf einen Vorschlag verzichtet werden, um den Codierer nicht zu einer falschen Codierung zu verleiten (vgl. Wettstein, 2014b: 36).

Der Vorteil halbautomatischer Analysen liegt einerseits in der Zeitersparnis des Codierers, der in den meisten Fällen nur noch eine Korrektur falscher Vorhersagen vornehmen muss. Andererseits wird in Fällen, in denen der Codierer eine andere Entscheidung getroffen hätte als das die Eingabemaske vorschlägt, eine intensive Auseinandersetzung mit der Entscheidung angestossen. Die resultierende systematische Verarbeitung des Textes und der widersprüchlichen Einschätzungen führt zu einer fundierten Entscheidung, da eine heuristische Verarbeitung vermieden wird (vgl. Wettstein, 2014b: 36).

Andererseits muss jedoch auch eingeräumt werden, dass eine halbautomatische Inhaltsanalyse Gefahren für die Qualität der Codierungen birgt. So kann ein zu hohes Vertrauen in die Technik die Codierer dazu verleiten, falsche Entscheidungen kritiklos anerkennen. Um dies zu vermeiden sollte bereits in der Codiererschulung Gewicht auf den Umgang der Codierer mit der Software gelegt werden. Die Codierer müssen sich der Bedeutung manueller Korrekturen für die Optimierung des Verfahrens bewusst werden, um aktiv nach Fehlern in den automatischen Vorhersagen zu finden. Zusätzlich kann die Übereinstimmung der Codierer mit den automatischen Vorhersagen kontinuierlich überprüft werden, um ungewöhnlich hohe Annahmequoten der Vorschläge zu identifizieren und die Codierer bei Bedarf zu Vorsicht zu ermahnen (vgl. Wettstein, 2014b: 36f).

Umsetzung

In *Angrist* wurde die halbautomatische Inhaltsanalyse bei der Entwicklung mitgedacht, wodurch die Einbindung automatischer Verfahren über das externe Programm *Aeglos* jederzeit möglich ist. Der Codierer bearbeitet den Text direkt am Bildschirm und kann, falls mehrere Analyseeinheiten in einem Text analysiert werden, diese farblich hervorheben und nacheinander bearbeiten. Das Programm merkt sich dabei jeweils die farblich hervorgehobene Stelle und den Absatz, in welchem

diese Stelle markiert wurde. Als Hilfestellung lassen sich in *Angrist* zentrale Begriffe der Analyse – beispielsweise die Namen von Akteuren oder wichtige Begriffe – automatisch farblich hervorheben.

Die Abfragetypen, über welche die Codierer ihre Codierentscheidungen eingeben, sind so angelegt, dass sie standardmässig den Text des aktuell bearbeiteten Absatzes, sowie die den Namen der betreffenden Variable an *Aeglos* senden, um nach einer automatischen Codierung zu fragen. Liefert das Programm ein Ergebnis, so kann dies in die Darstellung der Abfrage eingebunden werden. Liefert *Aeglos* kein Ergebnis, weil für die Variable kein Training oder keine Regel vorliegt, die eine Vorhersage ermöglichen würde, so wird der Codierer ohne Vorschläge nach seiner Entscheidung gefragt. Die Art und Weise, der Umsetzung von Vorschlägen ist je nach Eingabemodalität unterschiedlich:

Bei nominalen oder ordinalen Kategorien mit wenigen Ausprägungen wird die Vorhersage von *Aeglos*, falls sie eine hohe Wahrscheinlichkeit hat, korrekt zu sein, direkt als Vorschlag unterbreitet. Wie gut eine Vorhersage sein muss, um angezeigt zu werden, kann vom Forscher für jede Variable definiert werden. Standardmässig wird eine berechnete Wahrscheinlichkeit von über 70% erwartet.

Bei langen Listen, mit denen beispielsweise Themengebiete oder Argumente abgefragt werden können, werden die fünf Optionen mit der höchsten berechneten Wahrscheinlichkeit getrennt von den restlichen Elementen am Anfang der Liste dargestellt. Sie werden dabei nach ihrem jeweiligen Code und nicht nach ihrer Wahrscheinlichkeit sortiert, um den Codierer nicht zu verleiten.

Bei Listen von Akteuren können all jene Elemente aus der Liste ausgeschlossen werden, von welchen kein Teil im markierten Text auftritt. Wird also die Textstelle "Von Kanzelerin Merkel erfahren wir:" markiert, so werden alle Namen aus der Liste gelöscht, welche weder Von noch Kanzlerin noch Merkel enthalten. Dies lässt zwar neben der Auswahl "Angela Merkel" auch Namen wie "Ursula von der Leyen" stehen, die Liste wird jedoch stark reduziert. Falls sie zu stark eingeschränkt ist und der Codierer den richtigen Namen nicht findet, kann er mit einem Klick die gesamte Liste anzeigen.

Bei dichotomen Kategorien, für die der Codierer zutreffende Optionen mit einem Häkchen markieren muss, werden jene Häkchen gesetzt, für welche eine ausreichend hohe Wahrscheinlichkeit berechnet wurde.

Beim Einsatz dieser Umsetzung in Inhaltsanalysen sollte darauf geachtet, die Codierer nicht durch aufdringliche oder möglicherweise fehlerhafte Vorschläge zu verunsichern und in ihrer Arbeit zu behindern. Sollte sich für einzelne Kategorien herausstellen, dass sie für eine automatische Bestimmung ungeeignet sind, so sollte bei diesen Kategorien auf den Einsatz verzichtet werden.

Bereits eine teilweise Umsetzung der halbautomatischen Inhaltsanalyse – zum Beispiel durch die Reduktion von Akteurslisten – kann die Arbeit der Codierer erleichtern. Es müssen nicht zwingend alle Kategorien automatisch voranalysiert werden.

7.2. Datenaufbereitung

Die Aufbereitung und Visualisierung ist ein Arbeitsschritt, der bereits von Stuckardt (2000) als mögliches Einsatzgebiet von Computern in einer computerunterstützten Inhaltsanalyse identifiziert wurde (S. 26f). Dies gilt sowohl für die qualitative als auch für die quantitative Analyse, welche auf unterschiedliche Arten von Softwarelösungen profitieren können. Bei der qualitativen Analyse eignen sich Computerprogramme, um auf die untersuchten Textstellen zuzugreifen, Kategorien zu verwalten und deren Verknüpfungen im Textkorpus zu visualisieren (vgl. Stuckardt, 2000: 27f). In der quantitativen Analyse können Statistikprogramme eingesetzt werden, um Daten zu aggregieren, Werte zu recodieren und Indizes zu bilden. In diesem Abschnitt soll vor allem auf zwei besondere Anwendungen für quantitative Inhaltsanalysen eingegangen werden. Für spezielle Anwendungen der qualitativen Inhaltsanalyse sei auf die Arbeit von Mayring (2007: 100ff) und Stuckardt (2000: 25ff) verwiesen.

Die hier vorgestellten Verfahren, sowie weitere Verfahren zur Aggregation, Formatierung, Zusammenführung und Auswertung von Inhaltsanalysedaten wurden im Rahmen dieser Dissertation entwickelt und in *Nogrod* implementiert. Dieses Programm wurde speziell entwickelt, um Rohdaten von Inhaltsanalysen vor deren Import in ein Statistikprogramm zu bearbeiten und erfordert in der Anwendung keine Programmierkenntnisse. Die einfache Einbindung benutzerdefinierter Makros in dieses Programm erlaubt zudem die wiederholte Ausführung komplexer Formatierungsarbeiten oder Abfragen wie der Extraktion von Arbeitszeiten oder die Kombination von Informationen aus mehreren Tabellen zur Indexbildung. Die Dokumentation (siehe Anhang B) enthält deswegen auch einzelne projektspezifische Funktionen, die nur für die Daten der betreffenden Inhaltsanalysen funktionieren.

7.2.1. Identifikation von Mustern in Inhaltsanalysedaten

Falls bei quantitativen Inhaltsanalysen eine kleinteilige Messung über eine Vielzahl von Dummy-Variablen vorgenommen wird, wie an verschiedenen Stellen empfohlen wird (vgl. Matthes & Kohring, 2008; Semetko & Valkenburg, 2000; Lee & Maslog, 2005), so liegen die Daten am Ende der Datenerhebung als umfangreiche Dummy-Tabellen mit dichotomen Variablen oder Zähldaten vor. Als Forscher steht man dann vor der Herausforderung, in diesen Daten Muster zu erkennen, die sich vor dem Hintergrund des Forschungsinteresses inhaltlich interpretieren lassen. Als mögliche Methoden wurden Clusteranalyse (vgl. Matthes & Kohring, 2008), die Analyse latenter Klassen (vgl.

Matthes, 2007), eine Smallest Space Analysis (vgl. Miller & Riechert, 2001) oder explorative Faktoranalyse (vgl. Semetko & Valkenburg, 2000) empfohlen. Auch eine konfirmatorische Faktoranalyse kann eingesetzt werden, um theoretisch hergeleitete Muster in der Lösung zu überprüfen und gegebenenfalls die Messinvarianz in unterschiedlichen Zeitungen, Ländern oder Zeiträumen zu prüfen (vgl. Semetko & Valkenburg, 2000; Wirth, Wettstein, Reichel, & Kühne, 2013).

All diese Verfahren wurden jedoch für die Identifikation von Gruppen metrischer Variablen unter Annahme einer gauss'schen Verteilung optimiert und können bei der Analyse von Matrizen mit Zähldaten und vielen Nullen zu schlechten Ergebnissen führen (vgl. Wettstein, Submitted). Gleichzeitig muss bedacht werden, dass nicht alle erhobenen Elemente zu einer Gruppe gehören, und oftmals einzelne Variablen ausgeschlossen werden müssen, um die Resultate zu verbessern. Die explorative Arbeit, die dabei zu leisten ist, ist jedoch zeitaufwändig und erfordert viele kleine – oftmals spontane und schwer nachvollziehbare – Entscheidungen seitens des Forschers.

Mit meinem Beitrag (Submitted) zur explorativen Clusteranalyse (HECATE: Hierarchical Exploratory Cluster analysis with Automated Term Exclusion) stelle ich ein Verfahren vor, das in diesen Situationen die Mustererkennung deutlich erleichtern kann. Das Verfahren ist für die Analyse von Tabellen mit Zähldaten und vielen Nullen (Zero-inflated count data: vgl. Dobbie & Welsh, 2001) optimiert und schliesst automatisch Elemente aus der Lösung aus, die nicht zu einem homogenen Cluster gehören. Die Analyse eignet sich durch dieses Vorgehen sowohl zur Aufbereitung manuell erhobener Zähl- und Dummyvariablen, als auch zur Identifikation von Mustern in automatisch erhobenen Daten zum Auftreten einzelner Begriffe oder semantischer Kategorien in Texten.

In drei Studien habe ich in diesem Beitrag gezeigt, dass das Verfahren homogene, inhaltlich interpretierbare Cluster aus unterschiedlichen Datenquellen extrahieren kann, dass die Lösungen auch dann robust bleiben, wenn weitere Informationen hinzugefügt werden, dass zufällig auftretende Elemente als Rauschen ausgeschlossen werden und dass die Analyse von rein zufällig generierten Daten zu keinem sinnvollen Ergebnis führt. Diese Auswertungsmethode ist daher insbesondere für die Identifikation von Frames nach dem Paradigma von Matthes und Kohring (2008) einsetzbar.

7.2.2. Verlaufsanalyse zur Erforschung von Aufmerksamkeitsschüben

Bei Inhaltsanalysen, die ein Thema über einen langen Zeitraum erforschen, können in unregelmässigen Abständen kurze Phasen ausgemacht werden, in denen besonders viele Artikel oder Aussagen zum untersuchten Thema publiziert werden. Diese Aufmerksamkeitsschübe (vgl. Barabási, 2005) können unterschiedliche Auslöser und Gründe haben. So können plötzliche Ereignisse (vgl. Kepplinger, 2001; Brosius & Eps, 1995), politische Debatten (vgl. Downs, 1972) oder erfolgreiche

Medienkampagnen die öffentliche Aufmerksamkeit wecken und zu einer verstärkten Berichterstattung führen.

Mit der Analyse des Verlaufs der Medienaufmerksamkeit und der Fokussierung auf Unterthemen und Akteure habe ich (In Press) ein Verfahren vorgeschlagen, mit dem sich Aufmerksamkeitsschübe identifizieren und klassifizieren lassen. Ich konnte zeigen, dass sich die Fokussierung vor und während eines Aufmerksamkeitsschubes unterschiedlich verhält, wenn ein Ereignis, eine Debatte oder eine Kampagne das öffentliche Interesse geweckt haben. Diese Erkenntnis wurde bereits in Kapitel 4.1.2 eingesetzt, um Ereignisse und Debatten mittels eines einfachen automatischen Verfahrens zu bestimmen.

Bei der Datenaufbereitung selber durchgeführter quantitativer Inhaltsanalysedaten ist dieses Verfahren für die Identifikation von Ereignissen und Debatten unnötig kompliziert, da eine kurze Rückfrage an die Codierer (z.B: "Was war denn da im Juni los?") die gleichen Informationen in detaillierterer Form zugänglich macht. Dennoch kann dieses Verfahren bei Sekundäranalysen oder bei der Datenaufbereitung helfen, da es den Zeitpunkt der thematischen Fokussierung vor einer politischen Entscheidung, die Dauer einer Kampagne oder die Nachwirkungen eines Skandals nachvollziehbar sichtbar macht. Gerade wenn explorative Verfahren eingesetzt werden, um Muster im gemeinsamen Auftreten semantischer Elemente zu finden, bietet sich zudem eine vorgängige Identifikation besonderer Phasen der Berichterstattung an, in welchen ein Ereignis oder eine Debatte die Inhalte des öffentlichen Diskurses verändern.

Ein Beispiel, das ich in diesem Zusammenhang in beiden Publikationen (In Press; Submitted) anbringe, ist die Berichterstattung über das Thema der Arbeitslosigkeit in Grossbritannien im Herbst 2010. Durch die Ankündigung am 20. Oktober 2010, eine halbe Million Stellen im öffentlichen Dienst zu streichen, wurde der öffentliche Diskurs über Arbeitslosigkeit in Grossbritannien drastisch verändert. In der Verlaufsanalyse lässt sich bestimmen, wie lange das Ereignis die Berichterstattung dominierte und wann die Fokussierung auf die Stellenkürzungen begann. Trennt man die Berichterstattung in zwei Phasen (vor und nach der Ankündigung) auf, bevor man nach einem Muster in den Codierungen sucht, so erkennt man eine deutliche Veränderung im Framing der Debatte (vgl. Wettstein, Submitted). Dieser Wandel wäre bei einer Analyse der Muster über den gesamten Erhebungszeitraum verborgen geblieben.

Mit der longitudinalen Analyse der Verlaufs und der Clusteranalyse zur Identifikation von Mustern in einzelnen Phasen der Berichterstattung können kleinteilig erhobene Daten quantitativer Inhaltsanalysen optimal explorativ ausgewertet werden. Die Ergebnisse dieser Analysen können im Anschluss verwendet werden, um einerseits den Diskurs qualitativ zu beschreiben und andererseits

den einzelnen Texten Informationen zu ihrer Zugehörigkeit zu einem bestimmten Cluster oder einer Phase zuzuspielen. Beide Verfahren wurden in *Nogrod* implementiert und können mit diesem Programm über benutzerfreundliche Menüs ausgeführt werden.

7.3. Betreuung

Die Betreuung des Codierteams während der Inhaltsanalyse ist eine administrative Aufgabe, die vor allem zwischenmenschliche Kommunikation und regelmässige Konsultationen beinhaltet. Bei diesen Aufgaben sind elektronische Hilfsmittel – abgesehen von elektronischen Kommunikationskanälen – nur schlecht einzubinden. Dennoch gibt es auch in diesem Arbeitsschritt zwei wichtige Aufgaben, die mit der Einbindung geeigneter Software erleichtert werden können. Die erste Aufgabe ist die Zuweisung von Untersuchungsmaterial an die Codierer und die Sammlung der produzierten Daten. Die zweite Aufgabe ist die buchhalterische Verwaltung und Abrechnung der Arbeitszeiten.

Ein zusätzliches Einsatzgebiet von Computern in diesem Arbeitsschritt schlagen Wirth et al. (2015) mit der Überwachung der Randdaten von Codierern vor. Über diese Messungen ist es möglich, schadhaftes Codierverhalten rechtzeitig zu erkennen und bei Bedarf zu intervenieren.

7.3.1. Automatische Zuweisung von Texten und Sammlung von Daten

In Inhaltsanalysen mit mehreren Codierern und einem grossen Textkorpus ist es erforderlich, die Codierer regelmässig mit neuem Material zu versorgen. Gleichzeitig produzieren die Codierer jeweils unabhängig voneinander grosse Mengen an Daten, die regelmässig gesammelt werden müssen. Dauert diese Aufgabe pro Codierer auch nur wenige Minuten, ist bei umfangreichen Projekten mit mehr als 20 Codierern mit einem konstanten Arbeitsaufwand zu rechnen. Gerade die Auswahl von Texten aus dem Korpus ist bei grossen Projekten aufwändig, wenn eine zufällige Stichprobe gezogen werden soll. Hier muss auch darauf geachtet werden, dass einzelne Texte nicht gleichzeitig an zwei Codierer herausgegeben werden.

Eine computerunterstützte Lösung für dieses Problem wurde in der Inhaltsanalyse des NCCR-Moduls "*Populismus in Zeiten der Globalisierung und Mediatisierung*" entwickelt und realisiert. Mit mehr als 50 Codierern und aktuell über 90'000 Texten in 8 Sprachen und über 300 individueller Teilkorpora gilt diese Inhaltsanalyse zweifellos als ausserordentlich umfangreiches Projekt. Einen Überblick über die Arbeit aller Codierer, der Zuweisung von Texten aus allen Teilkorpora und den Codierfortschritt zu behalten, wäre mit manuell geführten Listen kaum möglich. Aus diesem Grund wurde über die Software, welche für *Angrist* entwickelt wurde, ein weiteres Dienstprogramm mit dem vorläufigen Namen *Statuscheck* entwickelt, das bei der Administration der Codierer helfen kann. Die Software greift dabei auf die einzelnen Tabellen der administrativen Datenbank des Moduls

(siehe Kapitel 4.2.1) zu, um die Sprachkenntnisse der Codierer, die Verfügbarkeit von Texten und die Organisation der Teilkorpora zu erfassen. Gleichzeitig greift sie auf die Verzeichnisse der einzelnen Codierer zu, um die Daten laufend zu sammeln und die Materialversorgung der Codierer zu prüfen.

In Abbildung 20 ist ein Screenshot aus dem Programm dargestellt. Rechts wird oben eine Liste aller Codierer angezeigt. Für jeden Codierer werden die noch verbleibenden Texte im Ordner und die Sprachkenntnisse aufgeführt. Codierer mit weniger als 20 restlichen Texten sind speziell markiert ("<<<<"), um auf ein erforderliches Eingreifen hinzuweisen. Unterhalb dieser Liste sind alle Teilkorpora (Samples) aufgeführt. Für jedes Teilkorpus sind die Gesamtanzahl von Texten, die Anzahl bereits zugewiesener, und codierter Texte sowie die geplante und realisierte Ausschöpfungsquote angegeben. Diese Zahlen helfen bei der Entscheidung, aus welchem Teilkorpus als nächstes geschöpft werden soll. Auf der linken Seite kann entschieden werden, aus welchem Teilkorpus der aktuell ausgewählte Codierer (in diesem Fall ein französischsprachiger Codierer) Texte erhalten soll. Sowohl die Zuweisung von Texten aus dem laufenden Reliabilitätstest als auch Texte aus den Teilkorpora, deren Sprache der betreffende Codierer beherrscht, stehen zur Auswahl.

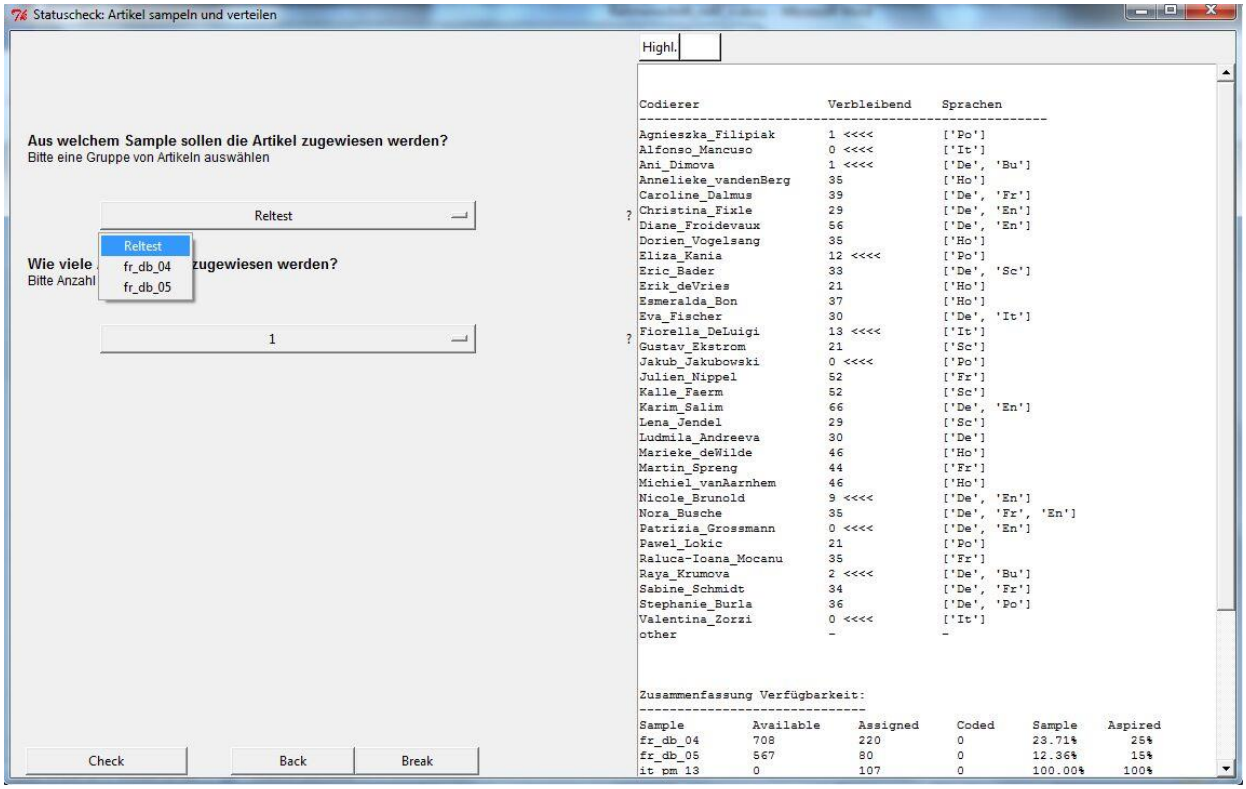


Abbildung 20: Ansicht des Programms Statuscheck zur Administration der Codierer. Rechts sind Informationen zu den Codierern und den Teilkorpora eingeblendet, links kann der Forscher einstellen, welche Texte welchem Codierer zugewiesen werden sollen.

Wird eine Auswahl getroffen, sucht das Programm die angeforderte Anzahl Texte zufällig aus dem angegebenen Teilkorpus aus und weist sie dem Codierer zu. Die Texte werden dafür automatisch vom zentralen Server in die Aufgabenliste des Codierers kopiert. Der Aufwand für die Projektleitung

wird durch dieses Dienstprogramm erheblich erleichtert, da eine manuelle Buchhaltung der Artikel und Codierer, sowie ein manuelles Kopieren und verschieben von Texten entfällt.

Bei jedem Neustart dieses Dienstprogrammes greift es auf die Daten aller Codierer zu, um ein Backup der Daten anzulegen und sie in zentralen Tabellen zu sammeln. Auf diese Weise werden täglich aktuelle Backups erstellt und die Gefahr, Daten zu verlieren, ist minimiert.

7.3.2. Messung von Randdaten der Erhebung

Werden die Daten durch die Codierer über ein GUI erfasst, so kann auch die Administration der Arbeitszeiten durch eine automatische Erfassung vereinfacht werden. Im Hintergrund der Erhebung kann das Programm die Zeiten notieren, zu welchen jeder Codierer aktiv ist und diese auf Abruf zur Verfügung stellen. Dies ist besonders hilfreich für die Abrechnung der Arbeitszeiten am Ende des Monats oder am Ende eines vereinbarten Zeitraums.

Zusätzlich zu den Arbeitszeiten pro Monat kann je nach Art der Erfassung auch die Zeit berechnet werden, die ein Codierer durchschnittlich für die Bearbeitung eines Textes benötigt. Anhand dieser Daten lassen sich die Hochrechnungen der Planungsphase gerade in der Frühphase der Erhebung noch korrigieren, um eine präzisere Schätzung des Arbeitsaufwandes erstellen zu können. Zudem können auffällig langsame oder schnelle Codierer erkannt werden, deren Verhalten darauf hinweist, dass sie bei ihrer Arbeit abgelenkt oder zu wenig vorsichtig sind. Mit diesen auffälligen Codierern kann ein Gespräch gesucht werden, um schädliche Einflüsse auf die Codierung frühzeitig auszuschliessen.

Schliesslich kann auch die Anzahl Texte, die im Durchschnitt in einer Codiersitzung bearbeitet werden, sowie die Tageszeit und Wochentage, zu welchen einzelne Codierer besonders produktiv sind, automatisch erfasst werden. Diese Informationen können die Projektleitung bei der Administration der Codierer unterstützen. So sollte zum Beispiel einem Codierer, welcher jeweils mittwochs und donnerstags viel arbeitet und im Durchschnitt 50 Texte an diesen beiden Tagen erfasst, jeden Mittwoch ausreichend viele Texte zur Verfügung gestellt werden.

7.3.3. Erfassung möglicher schadhafter Einflüsse

Mit der engmaschigen Beobachtung der Codierer bei ihrer Arbeit haben Wirth et al. (2015) eine weitere Anwendung für Computer in der Anwendungsphase der Inhaltsanalyse vorgeschlagen. Dabei werden Codierer einerseits bereits während der Schulung mit einem Fragebogen zu ausgewählten Persönlichkeitsmerkmalen befragt und während der Codierung anhand verschiedener Randdaten wie der Bearbeitungszeit, der aktiven Tageszeit und der Anzahl, Länge und Komplexität der Texte bei ihrer Arbeit beobachtet. Am Ende jeder Sitzung beantworten die Codierer einige Fragen zu ihrer

Arbeitsumgebung und allfälligen Hintergrundgeräuschen, Nebenaktivitäten und Ablenkungen (vgl. Wirth et al., 2015: 104f).

Die so erhobenen Daten können verwendet werden, um den Einfluss von Persönlichkeitsmerkmalen, Arbeitsroutinen und unterschiedlicher Arbeitsumfelder auf die Qualität und Effizienz der Codierer zu ermitteln. So konnten Wirth et al. (2015) zeigen, dass der kognitive Stil einen Einfluss auf die Qualität der Codierung komplexer Artikel haben kann, dass Codierer in den Abendstunden deutlich schneller arbeiten, und dass Codierer bereits nach den ersten 30-40 codierten Artikel eine gewisse Routine haben und ihre Effizienz kaum mehr steigern können. Überraschend konnte kein schädlicher Einfluss, des Geräusches von Fernsehgeräten oder Musik im Hintergrund auf die Reliabilität gefunden werden (S. 109f).

Durch eine eingehende Beschäftigung mit diesen Randdaten kann die Qualität laufender und künftiger Inhaltsanalyseprojekte verbessert werden. Indem der Forscher sich mit den Arbeitsroutinen und den möglichen schädlichen Einflüssen auf das Codierverhalten befasst, kann er ein Gefühl für mögliche Gefahren entwickeln und sie aktiv verhindern. So empfiehlt es sich aufgrund der Ergebnisse von Wirth et al. (2015), den Codierern die Nutzung von Musik während des Codierens nicht kategorisch zu verbieten.

Kapitel 8. Zusammenfassung

In den Kapiteln vier bis sieben habe ich in kurzen Zügen ein Methodeninventar für eine computerunterstützte Inhaltsanalyse skizziert, das gerade für die Kommunikationswissenschaft von Nutzen sein kann. Da bei einer computerunterstützten Inhaltsanalyse die finalen Codierentscheidungen noch immer durch den Menschen getroffen werden, opfert man mit keinem der hier vorgestellten Verfahren das Textverständnis und das Weltwissen, das die Codierer in eine Inhaltsanalyse einbringen. Stattdessen übernehmen Computer all jene aufwändigen kleinen Arbeiten im Prozess einer Inhaltsanalyse, mit denen sich der Forscher vor und während der eigentlichen Datenerhebung befassen muss. Die computerunterstützte Inhaltsanalyse wird damit nur in Ausnahmefällen (z.B. der halbautomatischen Inhaltsanalyse) zu einer Unterstützung für den Codierer. In den meisten Fällen unterstützen die Computer den Forscher oder die Projektleitung dabei, einen reibungslosen Ablauf der Inhaltsanalyse zu gewährleisten.

Nachdem in den vergangenen Kapiteln getrennt auf die einzelnen Verfahren einer computerunterstützten Inhaltsanalyse eingegangen wurde, werde ich nun abschliessend den gesamten Prozess einer maximal computerunterstützten Inhaltsanalyse skizzieren und den Nutzen einer vollständigen oder partiellen Umsetzung diskutieren.

8.1. Der Prozess der computerunterstützten Inhaltsanalyse

In Kapitel 2.5 stellte ich den allgemeinen Ablauf des Prozesses einer Inhaltsanalyse vor, in welchen in der Folge die einzelnen Verfahren eingeordnet wurden. Durch die gleichzeitige Einbindung aller diskutierten Verfahren in diesen Prozess lässt sich eine computerunterstützte Inhaltsanalyse skizzieren, in der die meisten Arbeitsschritte durch automatische Verfahren ersetzt oder ergänzt sind (siehe Abbildung 21). Wenn auch eine solche maximal computerunterstützte Inhaltsanalyse selten realisiert werden kann oder sollte, so bietet auch eine teilweise Übernahme einzelner Arbeitsschritte entscheidende Vorteile für die Effizienz oder Qualität der Analyse.

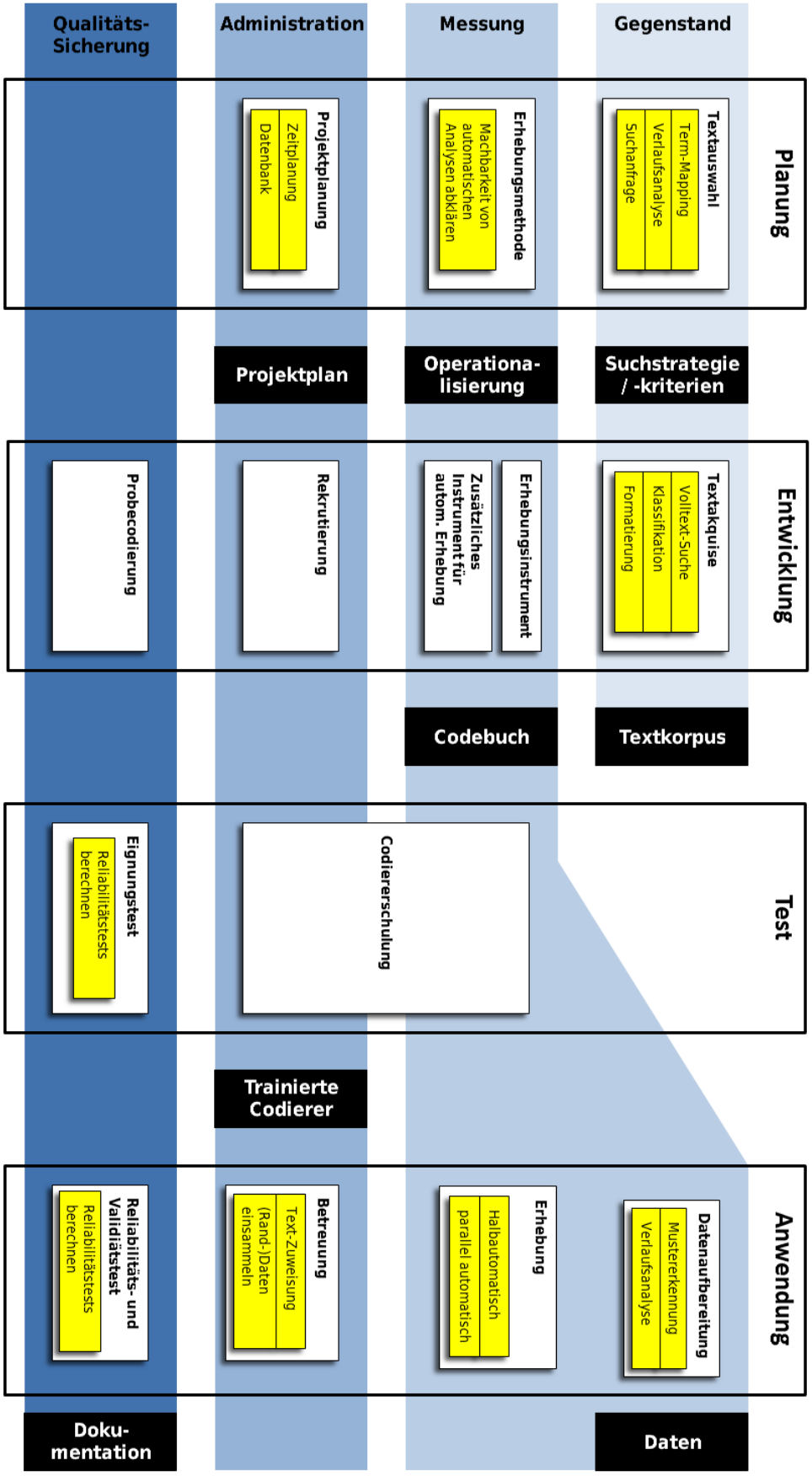


Abbildung 21: Prozessmodell der computerunterstützten Inhaltsanalyse bei Einbezug aller obengenannten Verfahren (gelbe Kästen).

Planungsphase

In der Planungsphase wird vor allem der Arbeitsschritt der Textauswahl durch automatische Verfahren unterstützt. Durch vorgängige explorative Analysen des Inhalts und Verlaufs der Berichterstattung und eine computerunterstützte Optimierung des Suchanfrage kann das relevante Teilkorpus effizient und ohne langwierige qualitative Vorstudien definiert werden. Gleichzeitig wird mit einer umsichtigen Projektplanung und der Anlage einer administrativen Datenbank die Grundlage für einen reibungslosen Ablauf der Inhaltsanalyse gelegt.

Entwicklungsphase

In der Entwicklungsphase wird traditionell die Textakquise mit Hilfe von Computern durchgeführt. Hier sind Zugänge zu digitalen Archiven und die trainierte Klassifikation von Texten zwei wichtige Hilfsmittel, um ein Textkorpus für die Inhaltsanalyse aufzubauen. Automatische Verfahren zur Formatierung der Texte unterstützen die Vorbereitung des Korpus.

Die Vorbereitung automatischer Verfahren für eine halbautomatische Inhaltsanalyse oder eine parallele automatische Datenerhebung kostet in der Entwicklungsphase Zeit und zusätzliche Ressourcen. Beides zahlt sich jedoch in der Anwendungsphase aus, wenn damit die Erhebung effizienter und weniger aufwändig gestaltet werden kann.

Testphase

In der Testphase lassen sich nur der Eignungstest der Codierer und ein Reliabilitätstest zur Prüfung der Eindeutigkeit des Codebuches automatisch durchführen. Für diese Anwendungen sind Computerprogramme bereits etabliert, wenn auch in vielen Fällen die Übereinstimmung über Tabellenkalkulations- und Statistiksoftware erhoben wird. Der Einsatz spezialisierter Programme für Reliabilitätsanalysen kann diesen Arbeitsschritt beschleunigen und vertiefte Einblicke in die Testergebnisse gewähren.

Anwendungsphase

In der Anwendungsphase lassen sich alle Arbeitsschritte durch automatische Verfahren ergänzen oder ersetzen. Durch eine Erfassung der Texte über Eingabemasken sowie halbautomatische und parallel durchgeführte automatische Erhebungen kann Zeit und Aufwand gespart werden. Gleichzeitig kann durch den geschickten Einsatz von Datenbanken und Software die Administration und die Qualitätsprüfung erleichtert werden. Für die traditionell durch Computer unterstützte Datenaufbereitung bieten sich mit innovativen Auswertungsmethoden neue Möglichkeiten zur

Identifikation von Mustern, besonderen Vorkommnissen und Veränderungen des Medieninhalts über die Zeit.

Kosten und Nutzen

Der zentrale Nutzen von Computerunterstützung in Inhaltsanalysen liegt in der Steigerung der Effizienz, Qualität und Handhabung insbesondere bei umfangreichen Inhaltsanalyseprojekten. Gleichzeitig muss jedoch bedacht werden, dass durch den Einbezug automatischer Verfahren zusätzlicher Arbeitsaufwand und unter Umständen auch ein zusätzlicher finanzieller Aufwand entsteht. Deswegen gilt für diese Analysen, dass der Nutzen und der Aufwand der einzelnen Verfahren für das jeweilige Forschungsprojekt geprüft werden sollten.

Dabei ist zu bedenken, dass sowohl der Nutzen der einzelnen Verfahren, als auch ihr Aufwand stark variiert. Die Pflege einer Datenbank oder die automatische Berechnung von Reliabilitätskoeffizienten sind ohne grossen Aufwand zu erreichen und leisten einen wichtigen Beitrag für die Inhaltsanalyse. Die automatische Erfassung einer einzelnen inhaltlichen Kategorie über komplexe syntaktische Extraktion ist jedoch kaum den Aufwand wert, wenn ein Mensch dieselbe Information in wenigen Sekunden verlässlicher codieren kann.

8.2. Beitrag dieser Dissertation

Die Idee einer computerunterstützten Inhaltsanalyse ist alles andere als neu. Dass Computer im Prozess einer Inhaltsanalyse eingesetzt werden können, um Texte zu beschaffen, Daten einzugeben, Reliabilitätstests durchzuführen und die Daten auszuwerten, wird bereits seit Jahren diskutiert (siehe insbesondere: Stuckardt, 2000: 25ff; Luzar, 2004: 147ff; Scharkow, 2012: 41ff). Dennoch müssen noch heute zwei Sachverhalte festgestellt werden: Erstens werden automatische Verfahren in der Kommunikationswissenschaft noch immer von einer Minderheit inhaltsanalytischer Studien verwendet (siehe Kapitel 2.1). Zwar werden Texte vermehrt automatisch gesucht, die Mehrheit der Studien arbeitet jedoch auch heute noch mit gedruckten Texten und manueller Codierung. Zweitens wird zwar in Methodenbüchern auf die Verfügbarkeit automatischer Methoden hingewiesen (vgl. Krippendorff, 2013: 208ff), dabei werden aber meist nur die automatischen Erhebungsmethoden oder Dienstprogramme zur Dateneingabe (vgl. Mayring, 2007: 100ff) erwähnt. Eine systematische Aufstellung unterschiedlicher Hilfsmittel, die ein Forscher tatsächlich für die Durchführung einer manuellen Inhaltsanalyse heranziehen kann, ohne auf menschliche Codierer verzichten zu müssen, fehlt bislang.

Genau diese Lücke habe ich in dieser Dissertation einerseits durch die Entwicklung neuer Verfahren für eine computerunterstützte Inhaltsanalyse, andererseits durch die systematische Aufarbeitung und Eingliederung der vorhandenen Verfahren zu schliessen versucht. Die Verfahren

zur explorativen Voranalyse (vgl. Wettstein, 2012), Datenerhebung (vgl. Wettstein, 2014) und Datenaufbereitung (vgl. Wettstein, in press; Wettstein, submitted), sowie die in dieser Rahmenschrift zusätzlich eingeführten und nicht publizierten Verfahren für die Unterstützung der Administration ergänzen das bis anhin verfügbare Methodeninventar der Kommunikationswissenschaft.

Die systematische Aufarbeitung und Gliederung automatischer Verfahren, die ich in dieser Rahmenschrift ausgeführt und abschliessend in Form einer maximal computerunterstützten Inhaltsanalyse zusammengefasst habe, soll nicht als Standard für Inhaltsanalysen, sondern als Bausatz für unterschiedliche inhaltsanalytische Projekte verstanden werden. Inhaltsanalysen, die umfangreich genug sind, um den Einsatz aller hier vorgestellten Verfahren zu rechtfertigen, sind in der Realität sehr selten. Dennoch bietet sich auch in kleinen Inhaltsanalysen ein Blick auf das hier vorgestellte Methodeninventar an, um einzelne hilfreiche Verfahren zu übernehmen.

8.3. Lücken im Methodeninventar

Das in Kapitel 8.1 zusammengefasste Methodeninventar der computerunterstützten Inhaltsanalyse weist an mehreren Stellen klar erkennbare Lücken auf. So wurde in dieser Dissertation keine Softwarelösung für die Entwicklung und Probecodierung des Erhebungsinstrumentes, die Rekrutierung von Codierern oder die Codiererschulung vorgeschlagen. Diese Lücken sind der Tatsache geschuldet, dass diese vier Arbeitsschritte aus einer Reihe von Entscheidungen bestehen, die nur der Forscher treffen kann und Probleme betreffen, die ein Computerprogramm nicht lösen kann.

Das heisst nicht, dass man sich bei diesen Arbeitsschritten nicht auf Ergebnisse automatischer Analysen inspirieren lassen soll. So sind explorative Voranalysen des Untersuchungsgegenstandes unter Umständen sehr hilfreich bei der Konzeption des Codebuches und beim Auffinden von Beispielen und Formulierungen für die einzelnen Kategorien (vgl. Salway, 2015). Verfahren, die ausschliesslich einen dieser vier Arbeitsschritte adressieren und den Forscher dabei unterstützten würden, sind mir aber bis heute nicht bekannt.

Es ist jedoch sehr gut möglich, dass auch für diese Arbeitsschritte in naher Zukunft Dienstprogramme entwickelt werden. Gleichzeitig ist es sehr gut möglich, auch für jene Arbeitsschritte weitere Softwarelösungen zu entwickeln, die heute bereits durch Computer unterstützt werden können. Das hier vorgestellte Methodeninventar möchte ich aus diesem Grund nicht als abschliessende Definition der Computerunterstützten Inhaltsanalyse verstanden wissen, sondern als methodologisches Framework, in das weitere Verfahren eingegliedert und weitere Arbeitsschritte integriert werden können.

Kapitel 9. Fazit

In den vergangenen 5 Jahren hatte ich das seltene Glück, aktiv in zwei Inhaltsanalysen mitzuarbeiten, die umfangreich genug waren, um jeden erdenklichen Aufwand für den Einbezug möglichst vieler automatischer Verfahren zu rechtfertigen. Zudem durfte ich beratend und aktiv an einigen weiteren interessanten Projekten mit kleinerem Umfang mitarbeiten und neue Verfahren kennen lernen, einbringen und wiederholt testen. Diese Erfahrung kommt für einen Programmierer mit einem abgeschlossenen Studium in Kommunikationswissenschaft und Computerlinguistik einer blühenden Spielwiese und einer konstanten Herausforderung gleich. Die hier vorgestellten Verfahren, sowohl die eigenhändig entwickelten als auch die vorgestellten, sind das Ergebnis einer jahrelangen, äusserst intensiven Auseinandersetzung mit der Frage, wie sich Computer in einer Inhaltsanalyse gewinnbringend einsetzen lassen, ohne dabei das Textverständnis und Weltwissen menschlicher Codierer zu opfern.

Der pragmatische Fokus meiner Auseinandersetzung mit diesem Thema führte nicht zu einer theoretischen Weiterentwicklung der Methode der Inhaltsanalyse, sondern zu einer praktisch orientierten Sammlung von Lösungen weit verbreiteter aber selten thematisierter Problemen bei der Durchführung von Inhaltsanalysen. Die resultierende kumulative Dissertation, die sich aus vier Beiträgen, drei funktionsfähigen Programmen und der vorliegenden Rahmenschrift zusammensetzt, soll in erster Linie die Verbreitung und Weiterentwicklung der computerunterstützten Inhaltsanalyse in der Kommunikationswissenschaft beflügeln.

Dass die Kommunikationswissenschaft – stärker als andere sozialwissenschaftliche Disziplinen wie die Politologie oder Soziologie – noch heute vornehmlich auf menschliche Codierer setzt und dem menschlichen Textverständnis und Weltwissen mehr Vertrauen entgegenbringt als einem Computerprogramm, hat durchaus seine Berechtigung. Noch heute sind auch die besten automatischen Verfahren nicht in der Lage, die Nachrichtenwerte eines Artikels, die Argumentation eines Politikers oder unterschiedliche Interpretationen eines Ereignisses auch nur annähernd so verlässlich zu erfassen wie ein Mensch. Computern diese Codierentscheidungen anzuvertrauen wäre deswegen kaum ratsam. In dieser Dissertation konnte ich jedoch zeigen, an wie vielen anderen Stellen im Prozess einer Inhaltsanalyse sich Computer gewinnbringend einsetzen lassen, um sowohl Forscher als auch Codierer in ihrer Arbeit zu unterstützen.

Literatur

Avraham, E., Wolfsfeld, G., & Aburaiya, I. (2000). Dynamics in the News Coverage of Minorities: The Case of the Arab Citizens of Israel. *Journal of Communication Inquiry, 24*(2), 117–133. doi:10.1177/0196859900024002002

Alexa, M. (1997). *Computer-assisted text analysis methodology in the social sciences* (ZUMA Arbeitsbericht No. 97/07). Mannheim. Retrieved from ZUMA website: http://www.gesis.org/fileadmin/upload/forschung/publikationen/gesis_reihen/zuma_arbeitsberichte/97_07.pdf

Barabási, A.-L. (2005). The origin of bursts and heavy tails in human dynamics. *Nature, 435*(7039), 207–211. doi:10.1038/nature03459

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research*, *3*(1), 993–1022.

Bloombaum, M. (1970). Doing Smallest Space Analysis. *Journal of Conflict Resolution, 14*(3), 409–416.

Bos, W., & Tarnai, C. (Eds.). (1996). *Waxmann-Wissenschaft*. *Computerunterstützte Inhaltsanalyse in den empirischen Sozialwissenschaften: Theorie, Anwendung, Software* (2. Aufl.). Münster: Waxmann.

Brosius, H.-B., & Haas, A. (2009). Auf dem Weg zur Normalwissenschaft. *Publizistik, 54*(2), 168–190. doi:10.1007/s11616-009-0034-0

Brosius, H.-B., & Eps, P. (1995). Prototyping through Key Events: News Selection in the Case of Violence against Aliens and Asylum Seekers in Germany. *European Journal of Communication, 10*(3), 391–412. doi:10.1177/0267323195010003005

Boydstun, A. E. (2008). *How Policy Issues become Front Page News*. Dissertation in Political Science, Pennsylvania State University. Ann Arbor: ProQuest.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement, 20*(1), 37–46. doi:10.1177/001316446002000104

Conway, M. (2006). The Subjective Precision of Computers: A Methodological Comparison with Human Coding in Content Analysis. *Journalism & Mass Communication Quarterly, 83*(1), 186–200.

Crawley, C. E. (2007). Localized Debates of Agricultural Biotechnology in Community Newspapers: A Quantitative Content Analysis of Media Frames and Sources. *Science Communication, 28*(3), 314–346.

David, C. C., Atun, J. M. L., & La Viña, A. G. (2010). Framing the population debate: A comparison of source and news frames in the Philippines. *Asian Journal of Communication, 20*(3), 337–353. doi:10.1080/01292981003802168

Dobbie, M. J., & Welsh, A. H. (2001). Modelling Correlated Zero-inflated Count Data. *Australian <html_ent glyph="@amp;" ascii="&"/> New Zealand Journal of Statistics, 43*(4), 431–444. doi:10.1111/1467-842X.00191

Donsbach, W., Laub, T., Haas, A., & Brosius, H.-B. (2005). Anpassungsprozesse in der Kommunikationswissenschaft. Themen und Herkunft der Forschung in den Fachzeitschriften „Publizistik“ und „Medien & Kommunikationswissenschaft“. *M&K,* 46–72. doi:10.5771/1615-634x-2005-1-46

Downs, A. (1972). Up and Down with Ecology: The "Issue-Attention Cycle". *Public Interest, 28,* 38–50.

Flick, U. (1990). Methodenangemessene Gütekriterien in der qualitativ-interpretativen Forschung. In J. B. Bergold & U. Flick (Eds.), *Forum für Verhaltenstherapie und psychosoziale Praxis: Vol. 14. Ein-Sichten. Zugänge zur Sicht des Subjekts mittels qualitativer Forschung* (2nd ed., pp. 247–262). Tübingen: DGVT.

Fretwurst, B. (2015). Reliabilität und Validität von Inhaltsanalysen: Mit Erläuterungen zur Berechnung des Reliabilitätskoeffizienten Lotus mit SPSS. In W. Wirth, K. Sommer, M. Wettstein, & J. Matthes

(Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 12. Qualitätskriterien in der Inhaltsanalyse* (pp. 177–204). Köln: Herbert von Halem Verlag.

Früh, W. (2009). *Inhaltsanalyse: Theorie und Praxis* (1st ed., Vol. 2501). Konstanz: UVK Verl.-Ges.

Gamson, W. A., & Modigliani, A. (1989). Media Discourse and Public Opinion on Nuclear Power: A Constructionist Approach. *American Journal of Sociology, 95*(1), 1–37.

Gerhards, J. (2003). Diskursanalyse als systematische Inhaltsanalyse: Die öffentliche Debatte über Abtreibungen in den USA und in der Bundesrepublik Deutschland. In R. Keller, A. Hirseland, W. Schneider, & W. Viehöver (Eds.), *Handbuch sozialwissenschaftliche Diskursanalyse: Vol. 2. Handbuch Sozialwissenschaftliche Diskursanalyse. Band 2: Forschungspraxis* (3rd ed., pp. 299–324). Wiesbaden: VS Verl. für Sozialwiss.

Günther, E., & Scharkow, M. (2014). Automatisierte Datenbereinigung bei Inhalts- und Linkanalysen von Online-Nachrichten. In K. Sommer, M. Wettstein, W. Wirth, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse* (pp. 111–126). Köln: von Halem.

Guttman, L. (1968). A General Nonmetric Technique for Finding the Smallest Coodrinate Space for a Configuration of Points. *Psychometrika, 33*(4), 469–506.

Hajer, M. A. (2003). Argumentative Diskursanalyse: Auf der Suche nach Koalitionen, Praktiken und Bedeutung. In R. Keller, A. Hirseland, W. Schneider, & W. Viehöver (Eds.), *Handbuch sozialwissenschaftliche Diskursanalyse: Vol. 2. Handbuch Sozialwissenschaftliche Diskursanalyse. Band 2: Forschungspraxis* (3rd ed., pp. 271–298). Wiesbaden: VS Verl. für Sozialwiss.

Hanke, C. (2003). Diskursanalyse zwischen Regelmässigkeiten und Ereignishaftem: am Beispiel der Rassenanthropologie um 1900. In R. Keller, A. Hirseland, W. Schneider, & W. Viehöver (Eds.), *Handbuch sozialwissenschaftliche Diskursanalyse: Vol. 2. Handbuch Sozialwissenschaftliche Diskursanalyse. Band 2: Forschungspraxis* (3rd ed., pp. 97–118). Wiesbaden: VS Verl. für Sozialwiss.

Hart, R. P. (2001). Redeveloping Diction: Theoretical Considerations. In M. D. West (Ed.), *Progress in communication sciences: Vol. 16. Theory, method, and practice in computer content analysis* (pp. 43–60). Westport, Conn: Ablex Publ.

Hayes, A. F., & Krippendorff, K. (2007). Answering the Call for a Standard Reliability Measure for Coding Data. *Communication Methods and Measures, 1*(1), 77–89.

Hellman, H. (2001). Diversity - An End in Itself?: Developing a Multi-Measure Methodology of Television Programme Variety Studies. *European Journal of Communication, 16*(2), 181–208. doi:10.1177/0267323101016002003

Jung, M. (2001). Dieskurshistorische Analyse: Eine linguistische Perspektive. In R. Keller, A. Hirseland, W. Schneider, & W. Viehöver (Eds.), *Handbuch sozialwissenschaftliche Diskursanalyse: Vol. 1. Handbuch Sozialwissenschaftliche Diskurs-Analyse. Band 1: Theorien und Methoden* (3rd ed., pp. 29–52). Opladen: Leske und Budrich.

Kahneman, D., Beatty, J., & Pollack, I. (1967). Perceptual Deficit during a Mental Task. *Science, 157*(3785), 218–219.

Keller, R. (2001). Wissenssoziologische Diskursanalyse. In R. Keller, A. Hirseland, W. Schneider, & W. Viehöver (Eds.), *Handbuch sozialwissenschaftliche Diskursanalyse: Vol. 1. Handbuch Sozialwissenschaftliche Diskurs-Analyse. Band 1: Theorien und Methoden* (3rd ed., pp. 113–144). Opladen: Leske und Budrich.

Kepplinger, H. M. (2001). Der Ereignisbegriff in der Publizistikwissenschaft. *Publizistik, 46*(2), 117–139.

Kepplinger, H. M., & Habermeier, J. (1995). The Impact of Key Events on the Presentation of Reality. *European Journal of Communication, 10*(3), 371–390. doi:10.1177/0267323195010003004

Keyling, T. & Jünger, J. (2013). *Facepager (Version, f.e. 3.3). An application for generic data retrieval through APIs*. Retrieved from https://github.com/strohne/Facepager

Kirilenko, A., Stepchenkova, S., Romsdahl, R., & Mattis, K. (2012). Computer-assisted analysis of public discourse: A case study of the precautionary principle in the US and UK press. *Quality & Quantity, 46*(2), 501–522. doi:10.1007/s11135-010-9383-z

Klingemann, H.-D. (Ed.). (1984). *Monographien sozialwissenschaftliche Methoden: Vol. 4*. *Computerunterstützte Inhaltsanalyse in der empirischen Sozialforschung*. Frankfurt/Main: Campus-Verl.

Kohring, M., & Matthes, J. (2002). The face(t)s of biotech in the nineties: How the German press framed modern biotechnology. *Public Understanding of Science, 11*(2), 143–154. doi:10.1088/0963-6625/11/2/304

Kolb, S. (2004). Verlässlichkeit von Inhaltsanalysedaten. Reliabilitätstest, Errechnen und Interpretieren von Reliabilitätskoeffizienten für mehr als zwei Codierer. *M&K,* 335–354. doi:10.5771/1615-634x-2004-3-335

Krippendorff, K. (2008). Systematic and Random Disagreement and the Reliability of Nominal Data. *Communication Methods and Measures, 2*(4), 323–338. doi:10.1080/19312450802467134

Krippendorff, K. (2013). *Content analysis: An introduction to its methodology* (3. ed.). Los Angeles, Calif.: Sage.

Kuckartz, U. (2007). *Einführung in die computergestützte Analyse qualitativer Daten* (2nd ed.). s.l: VS Verlag für Sozialwissenschaften (GWV).

Kuckartz, U. (1996). MAX für Windows: Ein Programm zur Interpretation, Klassifikation und Typpenbildung. In W. Bos & C. Tarnai (Eds.), *Waxmann-Wissenschaft. Computerunterstützte Inhaltsanalyse in den empirischen Sozialwissenschaften. Theorie, Anwendung, Software* (2nd ed., pp. 229–244). Münster: Waxmann.

Lee, S. T., & Maslog, C. C. (2005). War or Peace Journalism?: Asian Newspaper Coverage of Conflicts. *Journal of Communication, 55*(2), 311–329. doi:10.1111/j.1460-2466.2005.tb02674.x

Leskovec, J., Backstrom, L., & Kleinberg, J. (2009). Meme-Tracking and the Dynamics of the News Cycle. In J. Elder (Ed.), *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 497–506). New York, NY: ACM.

Lombard, M., Snyder-Duch, J., & Bracken, C. C. (2002). Content Analysis in Mass Communication: Assessment and Reporting of Intercoder Reliability. *Human Communication Research, 28*(4), 587–604.

Lombard, M., Snyder-Duch, J., & Bracken, C. C. (2010). *Practical Resources for Assessing and Reporting Intercoder Reliability in Content Analysis Research Projects*. Retrieved from http://matthewlombard.com/reliability/

Luzar, K. (2004). *Inhaltsanalyse von webbasierten Informationsangeboten: Framework für die inhaltliche und strukturelle Analyse*. Univ., Diss.--Münster (Westfalen), 2003. Norderstedt: Books on Demand.

Macnamara, J. (2005). Media content analysis: Its uses, benefits and best practice methodology. *Asia Pacific Public Relations Journal, 6*(1), 1–34. Retrieved from http://amecorg.com/wp-content/uploads/2011/10/Media-Content-Analysis-Paper.pdf

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, Mass: MIT Press.

Matthes, J. (2007). *Framing-Effekte: Zum Einfluss der Politikberichterstattung auf die Einstellungen der Rezipienten* (1st ed.). *Reihe Rezeptionsforschung: Vol. 13*. Baden-Baden: Nomos.

Matthes, J., & Kohring, M. (2008). The Content Analysis of Media Frames: Toward Improving Reliability and Validity. *Journal of Communication, 58*(2), 258–279. doi:10.1111/j.1460-2466.2008.00384.x

Mayring, P. (2007). *Qualitative Inhaltsanalyse: Grundlagen und Techniken* (9. Aufl., Dr. nach Typoskr). *UTB für Wissenschaft Pädagogik: Vol. 8229*. Weinheim: Beltz.

Miller, M., Andsager, J. L., & Riechert, B. P. (1998). Framing Candidates in Presidential Primaries: Issues and Images in Press Releases and News Coverage. *Journalism & Mass Communication Quarterly, 75*(2), 312–324.

Miller, M., & Riechert, B. P. (2001). Frame Mapping: A Quantitative Method for Investigating Issues in the Public Sphere. In M. D. West (Ed.), *Progress in communication sciences: Vol. 16. Theory, method, and practice in computer content analysis* (pp. 61–75). Westport, Conn: Ablex Publ.

Muhr, T. (1996). Textinterpretation und Theorieentwicklung mit ATLAS/ti. In W. Bos & C. Tarnai (Eds.), *Waxmann-Wissenschaft. Computerunterstützte Inhaltsanalyse in den empirischen Sozialwissenschaften. Theorie, Anwendung, Software* (2nd ed., pp. 245–260). Münster: Waxmann.

NCCR (2015). *Modul 2: Populismus in Zeiten der Globalisierung und Mediatisierung*. Retrieved from: http://www.nccr-democracy.uzh.ch/forschung/module2

Petty, R. E., Wells, G. L., & Brock, T. C. (1976). Distraction can enhance or reduce yielding to propaganda: Thought disruption versus effort justification. *Journal of Personality and Social Psychology, 34*(5), 874–884. doi:10.1037/0022-3514.34.5.874

Popping, R. (2000). *Computer-assisted text analysis*. *New technologies for social research*. London, Thousand Oaks, Calif: Sage Publications.

Popping, R. (2010). Some views on agreement to be used in content analysis studies. *Quality & Quantity, 44*(6), 1067–1078. doi:10.1007/s11135-009-9258-3

Quinn, K. M., Monroe, B. L., Colaresi, M., Crespin, M. H., & Radev, D. R. (2010). How to Analyze Political Attention with Minimal Assumptions and Costs. *American Journal of Political Science, 54*(1), 209–228. doi:10.1111/j.1540-5907.2009.00427.x

Roberts, C. W. (1989). Other than Counting Words: A Linguistics Approach to Content Analysis. *Social Forces, 68*(1), 147–177.

Rössler, P. (2010). *Inhaltsanalyse* (2nd ed.). Stuttgart: UVK-Verl.-Ges.

Salway, A. (2015, July). *The potential role of data-driven content analysis.* NCCR democracy 21, Universität Zürich, & Political Conflict in Europe. Symposium on new Frontiers of Automated Content Analyses in Social Sciences, Zürich. Retrieved from http://www.aca-zurich-2015.org/

Schäfer, M. S., Ivanova, A., & Schmidt, A. (2011). Globaler Klimawandel, globale Öffentlichkeit?: Medienaufmerksamkeit für den Klimawandel in 23 Ländern. *Studies in Communication/Media*, (1), 131–148.

Scharkow, M. (2012). *Automatische Inhaltsanalyse und maschinelles Lernen*. Dissertation Universität Hohenheim (1st ed.). Berlin: epubli GmbH.

Scharkow, M. (2015). *NewsClassifier*. Retrieved from https://github.com/mscharkow/newsclassifier

Scharkow, M., & Vogelgesang, J. (2015). Diagnose und Korrektur von Messfehlern in inhaltsanalytischen Daten. In W. Wirth, K. Sommer, M. Wettstein, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 12. Qualitätskriterien in der Inhaltsanalyse* (pp. 205–218). Köln: Herbert von Halem Verlag.

Schemer, C., Wirth, W., & Matthes, J. (2012). Value Resonance and Value Framing Effects on Voting Intentions in Direct-Democratic Campaigns. *American Behavioral Scientist, 56*(3), 334–352. doi:10.1177/0002764211426329

Schwab-Trapp, M. (2003). Methodische Aspekte der Diskursanalyse: Problem der Analyse diskursiver Auseinandersetzungen am Beispiel der deutschen Diskussion über den Kosovokrieg. In R. Keller, A. Hirseland, W. Schneider, & W. Viehöver (Eds.), *Handbuch sozialwissenschaftliche Diskursanalyse: Vol. 2. Handbuch Sozialwissenschaftliche Diskursanalyse. Band 2: Forschungspraxis* (3rd ed., pp. 171–197). Wiesbaden: VS Verl. für Sozialwiss.

Scott, W. A. (1955). Reliability of Content Analysis: The Case of Nominal Scale Coding. *Public Opinion Quarterly, 19*(3), 323–325.

Semetko, H. A., & Valkenburg, P. M. (2000). Framing European politics: A content analysis of press and television news. *Journal of Communication, 50*(2), 93–109. doi:10.1111/j.1460-2466.2000.tb02843.x

Sommer, K., Wettstein, M., Wirth, W., & Matthes, J. (2014). Zum Schattendasein der automatisierten Inhaltsanalyse: Ein Vorwort. In K. Sommer, M. Wettstein, W. Wirth, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse* (pp. 9–15). Köln: von Halem.

Stuckardt, R. (2000). *Qualitative Inhaltsanalyse durch Computer: Ein uneinlösbarer Anspruch?* Dissertation an der Johann Wolfgang Goethe Universität zu Frankfurt am Main. *Tenea Wissenschaft*. Berlin, Frankfurt (Main): Tenea.

Trilling, D. (2014). Weg vom manuellen Speichern: RSS-Feeds in ther automatisierten Datenerhebung bei Online-Medien. In K. Sommer, M. Wettstein, W. Wirth, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse* (pp. 73–89). Köln: von Halem.

van Atteveldt, W. (2015, May). *Using syntactic clauses for Social and Semantic Network Analysis.* International Communication Association. Annual Convention of the International Communication Association 2015, Puerto Rico, Puerto Rico. Retrieved from http://vanatteveldt.com/wp-content/uploads/ica2015_vanatteveldt_clauses.pdf

van Atteveldt, W., Kleinnijenhuis, J., & Ruigrok, N. (2008). Parsing, Semantic Networks, and Political Authority: Using Syntactic Analysis to Extract Semantic Relations from Dutch Newspaper Articles. *Political Analysis, 16*(4), 428–446. doi:10.1093/pan/mpn006

van Atteveldt, W., Ruigrok, N., Takens, J., & Jacobi, C. (2014). *Inhoudsanalyse met AmCAT*. Retrieved from http://vanatteveldt.com/wp-content/uploads/amcatbook.pdf

Wirth, W. (2001). Der Codierprozess als gelenkte Rezeption: Bausteine für eine Theorie des Codierens. In W. Wirth & E. Lauf (Eds.), *Inhaltsanalyse. Perspektiven, Probleme, Potentiale* (pp. 157–182). Köln: Halem.

Wirth, W., & Lauf, E. (2001). Vorwort. In W. Wirth & E. Lauf (Eds.), *Inhaltsanalyse. Perspektiven, Probleme, Potentiale* (pp. 7–12). Köln: Halem.

Wirth, W., Wettstein, M., Reichel, K., & Kühne, R. (2013). Äquivalenzprüfung als Standard in international vergleichenden Inhaltsanalysen. In T. Naab, D. Schlütz, W. Möhring, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 9. Standardisierung und Flexibilisierung als Herausforderungen der kommunikations- und publizistikwissenschaftlichen Forschung* (pp. 258–284). Köln: Herbert von Halem Verlag.

Wirth, W., Wettstein, M., Kühne, R., & Reichel, K. (2015). Theorie und Empirie des Codierens: Personelle und situative Einflussfaktoren auf Qualität und Quantität des Codierens bei der Inhaltsanalyse. In W. Wirth, K. Sommer, M. Wettstein, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 12. Qualitätskriterien in der Inhaltsanalyse* (pp. 96–118). Köln: Herbert von Halem Verlag.

Wedl, J., Herschinger, E., & Gasteiger, L. (2014). Diskursforschung oder Inhaltsanalyse: Ähnlichkeiten, Differenzen und In-/Kompatibilitäten. In J. Angermüller, M. Nonhoff, & E. Herschinger (Eds.), *Sozialtheorie: Vol. 1. Diskursforschung. Ein interdisziplinäres Handbuch* (pp. 537–563). Bielefeld: transcript.

Wessler, H. (1999). *Öffentlichkeit als Prozeß: Deutungsstrukturen und Deutungswandel in der deutschen Drogenberichterstattung*. Wiesbaden: VS Verlag für Sozialwissenschaften.

Wettstein, M. (2012a). Frame Adoption in Referendum Campaigns: The Effect of News Coverage on the Public Salience of Issue Interpretations. *American Behavioral Scientist, 56*(3), 318–333. doi:10.1177/0002764211426328

Wettstein, M. (2012b). Term-Mapping zur komparativen Analyse öffentlicher Debatten: Eine Anwendung der Smallest Space Analysis für Inhaltsanalysen. In B. Stark, M. Magin, O. Jandura, &

M. Maurer (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 8. Methodische Herausforderungen Komparativer Forschungsansätze* (pp. 138–157). Köln: Herbert von Halem Verlag.

Wettstein, M. (2014a). *Angrist 1.2: Documentation and Reference for the Coder Interface*. Retrieved from http://www.ipmz.uzh.ch/Abteilungen/Medienpsychologie/Recource/Angrist/ANGRIST_1-2-en.pdf

Wettstein, M. (2014b). "Best of Both Worlds": Die halbautomatische Inhaltsanalyse. In K. Sommer, M. Wettstein, W. Wirth, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse* (pp. 16–39). Köln: von Halem.

Wettstein, M. (Submitted). Identifying Clusters amid Noise: Hierarchical Exploratory Cluster Analysis with Term Exclusion. *Communication Methods and Measures*,

Wettstein, M. (In Press). Quantitative Ursachenbestimmung medialer Aufmerksamkeitsschübe. *Publizistik, 60*(3).

Wettstein, M., Wirth, W., & Reichel, K. (2015). Zum Problem der Mehrfachcodierung: Sind drei wirklich genug?: Eine systematische Fehleranalyse. In W. Wirth, K. Sommer, M. Wettstein, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 12. Qualitätskriterien in der Inhaltsanalyse* (pp. 219–236). Köln: Herbert von Halem Verlag.

Wüest, B., Clematide, S., Bünzli, A., Laupper, D., & Frey, T. (2011). Electoral Campaigns and Relation Mining: Extracting Semantic Network Data from Newspaper Articles. *Journal of Information Technology & Politics, 8*(4), 444–463. doi:10.1080/19331681.2011.567387

Young, L., & Soroka, S. (2012). Affective News: The Automated Coding of Sentiment in Political Texts. *Political Communication, 29*(2), 205–231. doi:10.1080/10584609.2012.671234

Zhu, J.-H. (1992). Issue Competition and Attention Distraction. *Journalism Quarterly, 69*(4), 825–836.

Anhänge

Anhang A: Dissertationsrelevante Publikationen

A1: Wettstein, M. (2012). Term-Mapping zur komparativen Analyse öffentlicher Debatten: Eine Anwendung der Smallest Space Analysis für Inhaltsanalysen. In B. Stark, M. Magin, O. Jandura, & M. Maurer (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 8. Methodische Herausforderungen Komparativer Forschungsansätze* (pp. 138–157). Köln: Herbert von Halem Verlag.

A2: Wettstein, M. (2014). "Best of Both Worlds": Die halbautomatische Inhaltsanalyse. In K. Sommer, M. Wettstein, W. Wirth, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse* (pp. 16–39). Köln: von Halem.

A3: Wettstein, M. (In Press). Quantitative Ursachenbestimmung medialer Aufmerksamkeitsschübe. *Publizistik, 60*(3).

A4: Wettstein, M. (Submitted). Identifying Clusters amid Noise: Hierarchical Exploratory Cluster Analysis with Term Exclusion. Submitted to: *Communication Methods and Measures*,

Anhang B: Dokumentation der Dienstprogramme

B1: Wettstein, M. (2014). *Angrist 1.2: Documentation and Reference for the Coder Interface*. Online verfügbar: http://www.ipmz.uzh.ch/Abteilungen/Medienpsychologie/Recource/Angrist/ANGRIST_1-2-en.pdf

B2: Wettstein, M. (2015). Nogrod 0.3(beta)*: Quick Tutorial*. Unveröffentlicht

B3: Wettstein, M. (2015). Aeglos 0.2(alpha)*: Quick Tutorial*. Unveröffentlicht

Anhang C: Akademischer Lebenslauf

Anhang A1: Term-Mapping

Veröffentlicht als:

Wettstein, M. (2012). Term-Mapping zur komparativen Analyse öffentlicher Debatten: Eine Anwendung der Smallest Space Analysis für Inhaltsanalysen. In B. Stark, M. Magin, O. Jandura, & M. Maurer (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 8. Methodische Herausforderungen Komparativer Forschungsansätze* (pp. 138–157). Köln: Herbert von Halem Verlag.

Term-Mapping zur komparativen Analyse öffentlicher Debatten. Eine Anwendung der Smallest Space Analysis für Inhaltsanalysen

[Bitte nicht aus diesem Manuskript zitieren. Manuskript zur Publikation: Wettstein, M. (2012). Term-Mapping zur komparativen Analyse öffentlicher Debatten: Eine Anwendung der Smallest Space Analysis für Inhaltsanalysen. In B. Stark, M. Magin, O. Jandura, & M. Maurer (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 8. Methodische Herausforderungen Komparativer Forschungsansätze* (pp.138–157). Köln: Herbert von Halem Verlag.]

Einführung

Die Vorbereitung von Inhaltsanalysen, Befragungen und Experimenten, bei denen ein bestimmter thematischer Fouks gewählt wird, bedürfen einer Auseinandersetzung mit dem Inhalt des gewählten Themas in der öffentlichen Diskussion, um relevante Themenbereiche, Akteure und Entwicklungen zu erkennen und bei der Erstellung des Instruments oder des Stimulus berücksichtigen zu können. Im Falle von Wahlen oder Abstimmungen, welche zeitlich und thematisch stark begrenzt sind, eignet sich hierzu eine Inhaltsanalyse der Pressemitteilungen beteiligter Akteure. Bei latenten Debatten über Themen wie Arbeitslosigkeit, Immigration, Globalisierung oder Umweltschutz ist es jedoch schwer, sich einen schnellen Überblick zu verschaffen, da sich die Debatten über mehrere Monate oder Jahre hinziehen, eine Vielzahl von Akteuren beteiligt ist und verschiedene Unterthemen von politischen Akteuren, Medien und der Bevölkerung diskutiert werden.

Eine Inhaltsanalyse, die eine solche Debatte abdeckt, muss sich in diesem Fall mit der gesamten Berichterstattung eines Landes über einen Zeitraum von mehreren Monaten auseinandersetzen. Um diesen Umfang an Texten effizient zu analysieren, bieten sich automatische Analyseverfahren an (Geis 2001: 320; Krippendorff 2004: 258-259), welche jedoch ebenfalls ein Grundwissen über den Gegenstand erfordern. In vielen Fällen ist eine manuelle Vorarbeit durch den Forscher wie die Definition von Analyseeinheiten und Kategorien nötig (Früh 2007: 287; Kuckartz 2007: 24). Zudem werden Sprachressourcen wie Lexika oder Thesauri benötigt, was die automatische Inhaltsanalyse von der Verfügbarkeit dieser Ressourcen in der gewünschten Sprache abhängig macht (Früh 2007: 287).

Das hier vorgestellte Verfahren des Term-Mapping bietet eine Möglichkeit, sich mit geringem Aufwand eine Übersicht über eine latente Debatte in den Medien zu verschaffen, die keiner Vorkenntnis der Themen und Akteure bedarf und unabhängig von der Sprache der Berichterstattung eingesetzt werden kann. Es eignet sich damit vor allem für die Vorbereitung von Befragungen oder Inhaltsanalysen, welche parallel in verschiedenen Ländern oder Sprachregionen durchgeführt werden, sowie für explorative Vergleiche einzelner Debatten in unterschiedlichen Zeiträumen, Ländern oder Regionen.

Aufbau des Term-Mapping

Das Term-Mapping-Verfahren zur oberflächlichen Analyse einer latenten Debatte verfolgt zwei Ziele. Zum einen sollen die zentralen Schlüsselbegriffe der Debatte identifiziert werden, welche neben häufig verwendeten Substantiven auch die Namen der in der Debatte vorkommenden Akteure sowie zentrale Verben und Adjektive umfassen.

Zum anderen wird das gemeinsame Auftreten der Schlüsselbegriffe der Debatte in der gesamten Berichterstattung – also auch in Artikeln ohne direkten Themenbezug – analysiert, um die Beziehungen zwischen den Schlüsselbegriffen in einer übersichtlichen Zusammenfassung darzustellen. Für diese Darstellung wird auf eine Methode zur multidimensionalen Skalierung, die Smallest Space Analysis (SSA), zurückgegriffen, welche es ermöglicht, die Beziehungen zwischen Paaren einer beliebig grossen Menge an Elementen als zwei- oder dreidimensionale Grafik zu veranschaulichen.

In diesem Beitrag soll zunächst die Methode der SSA und das Vorgehen beim Term-Mapping erläutert werden, um im Anschluss anhand eines konkreten Beispiels die Möglichkeiten und Ergebnisse dieses Verfahrens zu demonstrieren sowie dessen Anwendungsfelder und mögliche Modifikationen zu diskutieren.

Smallest Space Analysis

Die Smallest Space Analysis (SSA) wurde von Louis Guttmann (1968) erstmals publiziert. Er beschrieb sie als ein Verfahren, um eine Menge von Elementen, die eine Beziehung zueinander haben, auf möglichst kleinem Raum darzustellen (Guttmann 1968: 469). Die Elemente können dabei soziale Gruppen, Menschen, Ideen, Begriffe, Farben oder jede Form von Eigenschaften sein und ihre Beziehungen untereinander können Korrelationen, Gemeinsamkeiten oder Wahrscheinlichkeiten sein. Die Anzahl der Elemente ist nur durch die Rechenleistung (Bloombaum 1970: 409) und durch die Lesbarkeit des Outputs (Miller/Riechert 2001: 66) beschränkt. Das Verfahren ist damit aussergewöhnlich vielseitig und kann in verschiedenen Bereichen, wie hier zur Inhaltsanalyse, adaptiert werden.

Der kleinste Raum, der dem Verfahren seinen Namen gibt, bezieht sich dabei auf die Anzahl von Dimensionen, die benötigt werden, um die Beziehungen zwischen N Elementen mit einer möglichst geringen Verzerrung darzustellen. Dieser Raum wird durch schrittweise Reduktion der Dimensionen einer nicht verzerrten Repräsentation der Daten gefunden. Aus einer symmetrischen NxN-Distanzmatrix, welche als Eingabe für eine SSA erster Ordnung benötigt wird, wird dafür zunächst eine N-dimensionale Repräsentation ohne Verzerrungen errechnet. Dieses geometrische

Repräsentationsmodell wird nun schrittweise um jeweils eine Dimension reduziert, wobei auf eine möglichst schwache Verzerrung des Modells geachetet wird. Die Dimensionsreduktion läuft dabei so lange, bis die Verzerrung, welche in jedem Schritt berechnet und ausgewiesen wird, von einer Repräsentation zur nächsten stark ansteigt. Im Idealfall wird das Verfahren bis auf zwei oder drei Dimensionen fortgeführt, um die Repräsentation grafisch ausgeben zu können.

Das Ergebnis der Dimensionsreduktion ist damit ein zwei- oder dreidimensionaler Scatterplot, in welchem die Distanzen zwischen den einzelnen Elementen mit einer möglichst geringen Verzerrung repräsentiert sind. Wie Bloombaum (1970) es ausdrückte, sind «the results achieved [..] directly and intuitively interpretable by relatively untutored persons» (ebd.: 409). Dies liegt zum einen an der Tatsache, dass Elemente mit engen Beziehungen oder starken anfänglichen Korrelationen nahe beieinander liegen und gegebenenfalls optische, inhaltlich sinnvolle Cluster bilden. Zum anderen lässt sich über die globale Verteilung der Punkte des Plots schnell erkennen, wie viele Cluster durch die Elemente gebildet werden und wie diese zusammenhängen.

In der Kommunikationswissenschaft wurde diesem Verfahren bislang noch kaum Beachtung geschenkt. Erstmals angewandt wurde es in zwei Studien, welche das Auftreten von Frames in der Darstellung von politischen Kandidaten (Miller/Andsager/Riechert 1998) und in einer lokalen Diskussion über Umweltschutz (Miller/Riechert 2001) untersuchten. Diese definierten Listen von Schlüsselworten, welche auf die Verwendung einzelner Frames hinwiesen und stellten die Kollokation von Frames, Akteuren und Medien in einer dreidimensionalen SSA dar.

Da eines der Ziele des Term-Mapping gerade in der Identifikation von Unterthemen einer Debatte und der Schlüsselworte dieser Teildebatten besteht, kann hier jedoch keine Liste von Schlüsselworten und Akteuren vordefiniert werden. Stattdessen werden die Begriffe, welche in der Debatte eine Schlüsselrolle einnehmen, deduktiv aus der Berichterstattung gewonnen. Das Vorgehen gliedert sich dabei in fünf einzelne Schritte, in denen aus allen Begriffen in einer Menge von debattenrelevanten Artikeln jene extrahiert und analysiert werden, die für die Debatte zentral sind.

Vorgehen

Der Grundgedanke des Term-Mapping besteht darin, aus allen Worten einer Debatte eine Liste von N Schlüsselbegriffen zu extrahieren, die häufig im Zusammenhang mit zentralen Begriffen genannt werden, und die Beziehung unter ihnen zu analysieren. Die Auswahl erfolgt dabei blind, hängt also nicht vom Vorwissen des Forschers ab. Gleichsam hängt die Beziehung nicht von semantischen Verbindungen der Begriffe ab, sondern wird über die Wahrscheinlichkeit des gemeinsamen Auftretens berechnet.

Eine weitere Eigenheit, welche dieses Verfahren von vorgängigen Entscheidungen und Informationen unabhängig macht, ist die Grösse des Korpus, in dem die Analyse stattfindet. Das gemeinsame Auftreten der Begriffe wird nicht nur in einer Auswahl von Artikeln im Kern der Debatte untersucht, sondern in der gesamten Berichterstattung. Auf diese Weise werden von Unterthemen, die sich nur schwach mit der Kerndebatte überschneiden, alle Aspekte anstatt nur der Überschneidungen dargestellt. Sie lassen sich damit auch unabhängig von der Kerndebatte beschreiben.

Vorgehen bei der Begriffssuche

Um eine Liste von Schlüsselworten zu finden, ist zunächst die Definition der Debatte über einen bis zu drei zentralen Kernbegriffen erforderlich. Mit Hilfe dieser zentralen Begriffe (z.B: 'Arbeitslosigkeit' / 'Arbeitslose*' für die Arbeitslosigkeitsdebatte oder 'Immigration' / 'Einwanderung' für die Immigrationsdebatte) kann in einem ersten Schritt eine Stichprobe von 200 bis 300 Artikeln aus einer Artikeldatenbank gezogen werden, in denen über die Debatte berichtet wird.

Den Ausgangspunkt für die anschliessende Suche nach Schlüsselbegriffen bildet die komplette Liste von Worten, die in dieser ersten Stichprobe verwendet werden. Diese Liste muss zunächst reduziert werden, indem besonders seltene Worte und alle Präpositionen, Artikel, Zahlworte und Hilfsverben gestrichen werden. In flexionsreichen Sprachen bietet es sich zudem an, Gruppen von Worten, die sich nur in ihrer Endung unterscheiden, zusammenzufassen.

Diese Reduktionen führen schliesslich zu einer Liste mit 1.500 bis 2.500 Worten, die für die Debatte potentiell relevant sind. Die Liste enthält auch die Namen von Akteuren, die in der Debatte genannt werden und wie alle anderen Begriffe in die Analyse eingehen. Worte ohne Debattenbezug, wie die Namen von Journalisten oder bei der manuellen Reduktion nicht erkannte Präpositionen und Hilfsverben, sind vorläufig ebenfalls in der Liste enthalten, können im dritten Schritt aber erkannt und ausgeschlossen werden.

Vorbereitung des Korpus

Als Textkorpus für die Analyse dient die gesamte Berichterstattung eines Landes oder einer Zeitung über einen bestimmten Zeitraum. Bei Bedarf können Einschränkungen bezüglich des Ressorts oder der Art der Artikel gemacht werden, um beispielsweise Leserbriefe, Kommentare oder debattenfremde Ressorts wie die Sportberichterstattung auszuklammern. Bei grossen Zeiträumen empfiehlt sich zudem eine repräsentative Stichprobe der Artikel zu ziehen, um auf ein Korpus von nicht mehr als 20.000 bis 30.000 Artikeln zu kommen. Diese Obergrenze des Umfangs ist jedoch abhängig von der Rechenleistung der verwendeten Computer und der effizienten Implementierung

der folgenden Arbeitsschritte, wodurch sie sich in den kommenden Jahren nach oben verschieben wird.

Um den Umfang dieses Korpus für die folgenden Arbeitsschritte zu reduzieren, kann jeder Artikel gekürzt werden, indem alle Worte, die nicht zu den ca. 2000 potentiell relevanten Begriffen zählen, entfernt werden. Diese Worte sind für die folgenden Arbeitsschritte nicht mehr von Bedeutung, würden die Suche nach relevanten Begriffen in den Artikeln jedoch erheblich verlangsamen. Die Erfahrung über mehrere Debatten zeigte, dass durch diese Kürzung der Artikel die Textmenge um 80 bis 90 Prozent reduziert werden kann.

Identifikation der Schlüsselworte

Im nächsten Schritt wird die Liste der potentiell relevanten Begriffe auf Basis des Korpus weiter reduziert, um eine Gruppe von 100 bis 200 Schlüsselworten für die SSA zu finden. Zu diesem Zweck wird für jeden der potentiell relevanten Begriffe (W) das gemeinsame Auftreten mit den bereits eingangs definierten Kernbegriffen (K) über drei verschiedene Indikatoren gemessen.

Das erste Mass P(W|K) drückt die bedingte Wahrscheinlichkeit aus, dass W in einem Artikel auftritt, in welchem K genannt wird. Analog drückt P(K|W) die Wahrscheinlichkeit aus, dass ein Kernbegriff in einem Artikel auftritt, in dem W genannt wird. Der dritte Indikator ist das Produkt der beiden Wahrscheinlichkeiten und dient als symmetrisches Mass des gemeinsamen Auftretens zweier Begriffe (siehe Abbildung 1, Gl. 1-2).

Jene Begriffe, für welche P(W|K) deutlich grösser als P(K|W) ist, können aus der Liste entfernt werden, da es sich dabei mit hoher Wahrscheinlichkeit um zu allgemeine Begriffe, Namen von Journalisten oder andere Worte handelt, die in den meisten Artikeln vorkommen und nicht debattenspezifisch sind. Ebenso können Begriffe entfernt werden, bei denen das Produkt der beiden Wahrscheinlichkeiten niedrig ist, und die damit nicht oft gemeinsam mit den Kernbegriffen auftreten. Der Grenzwert für das gemeinsame Auftreten mit Kernbegriffen sollte dabei so gewählt werden, dass nur 100 bis 200 Schlüsselworte auf der Liste verbleiben. Diese Einschränkung ist notwendig, um die berechnete Darstellung übersichtlich und lesbar zu halten.

<Abbildung 1: Berechnung des gemeinsamen Auftretens und der Distanz zwischen Begriffen. Ngemeinsam: Anzahl gemeinsames Auftreten der beiden Begriffe; Nw1/Nw2: Totale Anzahl des Auftretens der Begriffe

Anmerkung für den Setzer: Hoch/Tiefstellungen aus den Formeln für die Legende übernehmen.>

Smallest Space Analysis

Für die Liste von 100 bis 200 Schlüsselworten wird in einem abschliessenden Schritt die symmetrische Distanzmatrix errechnet, welche als Ausganspunkt der SSA dienen soll. Dazu wird das Produkt der bedingten Wahrscheinlichkeiten paarweise für alle Schlüsselworte, inklusive der vordefinierten Kernbegriffe, berechnet. Begriffe, welche immer gemeinsam auftreten, haben nach dieser Rechnung ein gemeinsames Auftreten von 1. Bei nie gemeinsam auftretenden Worten ist der Wert genau 0.

Um die so entstehende Korrelationsmatrix der Begriffe in eine Distanzmatrix als Eingabe für die SSA zu überführen, wird der negative Logarithmus des gemeinsamen Auftretens berechnet. Immer gemeinsam auftretende Begriffe erhalten damit die Distanz 0. Für Begriffe, welche nie gemeinsam auftreten, ist die Distanz nicht definiert, kann jedoch durch die Annahme festgelegt werden, dass bei der doppelten Samplegrösse ein zufälliges gemeinsames Auftreten erwartet werden kann. Die Distanz ist damit durch Gleichung 3 in Abbildung 1 gegeben.

Die Distanzmatrix kann nun in einem Statistikprogramm als Eingabe für eine SSA verwendet werden. In der unten vorgestellten Pilotstudie wurde das Paket ‚smacof' für R verwendet, mit welchem sowohl zwei- als auch dreidimensionale Lösungen dargestellt werden können (DeLeeuw/Mair 2009).

Interpretation

Die Interpretation der Darstellung einer Distanzmatrix durch SSA ist ohne Vorkenntnisse der Debatten und ohne Kenntnis der Berechnungen leicht möglich. Bereits die Form des Scatterplots lässt Schlüsse auf die Form der Debatte zu, insbesondere auf ihre Homogenität, die Anzahl der getrennt diskutierten Unterthemen und die Nähe einzelner Unterthemen in der Berichterstattung.

Da die Nähe zwischen einzelnen Begriffen im Scatterplot die Wahrscheinlichkeit ihres gemeinsamen Auftretens in Artikeln ausdrückt, werden Gruppen von Begriffen, die oft gemeinsam vorkommen, als Ballungspunkte oder Cluster in der Menge sichtbar. Die einzelnen Cluster stehen damit für Unterthemen, in welchen die Schlüsselworte der Debatte vorkommen.

Die einzelnen Cluster können zum einen über die enthaltenen Begriffe inhaltlich interpretiert werden, zum anderen gibt ihre Gestalt Aufschluss über die Form der Debatte. Liegen die einzelnen Unterthemen im Scatterplot nahe beieinander, weist dies auf ein häufiges gemeinsames Auftreten der Begriffe zu beiden Themen hin. Eine grosse Distanz zwischen Unterthemen zeigt eine geringe inhaltliche Überschneidung an.

Ein weiterer Indikator für die Form der Debatte oder eines Unterthemas ist die Anordnung der einzelnen Begriffe innerhalb eines Clusters oder des gesamten Scatterplots. Sind sie um wenige zentrale Begriffe eng gruppiert, handelt es sich um eine homogene Debatte, bei der die einzelnen

Schlüsselworte nicht nur paarweise gemeinsam auftreten, sondern jeweils in ähnlichen Gruppen. Bei lockeren Clustern kann dagegen davon ausgegangen werden, dass es sich um heterogene Diskussionen handelt, bei denen die Begriffe zwar paarweise vorkommen, jedoch selten mit allen anderen Begriffen des Clusters. Durch die SSA wird in diesem Fall die Distanz zwischen den Begriffen innerhalb des Clusters gewahrt und es entsteht keine dichte Ballung von Begriffen.

In Abbildung 2 sind drei idealtypische Verteilungen des Scatterplots dargestellt. Abbildung 2a zeigt eine homogene Debatte über ein einzelnes Thema mit vereinzelten Begriffen, die nur mit einem Teil der Schlüsselworte gemeinsam auftreten und dadurch ausserhalb des Clusters liegen. In Abbildung 2b ist eine Debatte mit zwei Unterthemen mit unterschiedlicher Homogenität dargestellt. Das auf der linken Seite dargestellte Unterthema enthält Begriffe, die seltener in Gruppen vorkommen, jedoch paarweise gemeinsam auftreten können. Das rechts dargestellte Unterthema wird hingegen meist mit derselben Gruppe von Schlüsselworten diskutiert. In Abbildung 2c lässt sich eine gesamthaft etwas heterogene Debatte erkennen, die sich in vier Unterthemen aufteilen lässt.

Obschon die SSA eine Darstellung in einem zwei- oder dreidimensionalen Raum liefert, sind diese Dimensionen nicht als latente Faktoren interpretierbar. Anders als die Darstellung eines Faktorladungsdiagrammes lässt sich die Lösung einer SSA beliebig rotieren und skalieren, ohne das Resultat zu verändern, so lange die Distanz der einzelnen Elemente gewahrt bleibt. Die Lage der einzelnen Elemente ist damit nur hinsichtlich der Distanz zu den anderen Elementen, nicht über seine absolute Lage im Koordinatensystem zu interpretieren.

<Abbildung 2: Idealtypische Darstellungen des SSA Plots bei homogenen Debatten (a), getrennten Unterthemen (b) und verbundenen Unterthemen (c)>

Aufwand

Die einzelnen Schritte des Term-Mapping sind mit unterschiedlichem zeitlichem Aufwand verbunden. Den grössten Anteil hat dabei die Stichprobenziehung und Beschaffung der Artikel und die manuelle Reduktion der Liste potentiell relevanter Begriffe, die in Punkt 4.1 und 4.2 geschildert sind. Für beide Schritte sind pro Sprache mehrere Stunden einzuplanen. Die Aufbereitung des Korpus und die Berechnung des gemeinsamen Auftretens ist jedoch mit wenig Aufwand verbunden und hängt allein von der effizienten Implementierung dieser Arbeitsschritte ab. In der unten vorgestellten Pilotstudie wurden Python-Skripte eingesetzt, welche jeweils fünf bis zehn Minuten für die Berechnung einer Distanzmatrix und ihre Überführung in R-Syntax benötigten.

Fallbeispiel: Debatten in Europa

Das Verfahren des Term-Mapping wurde erstmals in der Vorbereitung einer internationalen Studie angewandt, in welcher die Entstehung, mediale Darstellung und Wirkung von öffentlichen Debatten in mehreren europäischen Ländern untersucht werden sollte. Die Studie sollte Interviews mit Politikern und Medienschaffenden sowie eine detaillierte Inhaltsanalyse und eine repräsentative Panel-Befragung der Bevölkerung in sechs Ländern (D, F, I, CH, GB, DK) umfassen (vgl. NCCR 2011).

Die Suche nach einer latenten Debatte, welche in allen Ländern mit vergleichbaren Fragen zu ähnlichen Unterthemen, politischen Massnahmen und Ereignissen untersucht werden kann, gestaltete sich in diesem Fall schwer, da eine konstante Berichterstattung über ähnliche Themen eine Voraussetzung für die Vergleichbarkeit der Ergebnisse in einzelnen Ländern ist. Um das zu gewährleisten, wurde die Berichterstattung über neun Monate in allen beteiligten Ländern zu drei in Frage kommenden Debatten (Arbeitslosigkeit, Immigration, Islam) mittels Term-Mapping analysiert.

Als Korpus wurde über künstliche Wochen in jedem Land eine repräsentative Stichprobe der in Lexis Nexis verfügbaren Zeitungsartikel zwischen August 2009 und Mai 2010 gezogen. Die Anzahl der Artikel lag dabei in jedem Land zwischen 15.000 und 20.000. Für die SSA wurden die Schlüsselworte auf jeweils 50 bis 70 Begriffe mit dem höchsten gemeinsamen Auftreten mit den Kernbegriffen (Pw12 > .03 <<Anmerkung an den Setzer: Tiefstellung aus Abb.1 übernehmen>>) reduziert und eine Reduktion bis auf zwei Dimensionen gerechnet, um die Lesbarkeit der Ausgabe zu gewährleisten.

Islam

Für die Debatte um den Islam wurden die Kernbegriffe ‚Islam' und ‚Islamisierung' in allen fünf Sprachen definiert. Die Ergebnisse der SSA sind in Abbildung 3 für die Schweiz und Grossbritannien exemplarisch angegeben. Bereits bei einer oberflächlichen Analyse fallen Unterschiede im inhaltlichen Fokus und in der Gewichtung der Unterthemen sowie bezüglich der Heterogenität der Debatten auf. Die deutlichen Unterschiede zwischen den einzelnen Ländern machen die Islamdebatte in Europa zu einem geeigneten Fallbeispiel für komparative Analysen durch SSA. Im Folgenden werden die Charakteristika der Debatte in den einzelnen Ländern, welche durch Term-Mapping gewonnen wurden, beschrieben, um das Vorgehen bei einer komparativen Analyse zu illustrieren.

<Abbildung 3: Smallest Space Analysis der Islamdebatte in der Schweiz (a) und Grossbritannien (b). Die umrahmten Begriffscluster weisen auf Unterthemen der Debatte hin, welche inhaltlich und strukturell interpretiert werden können>

Schweiz

In der Schweiz sind im Untersuchungszeitraum drei relevante, voneinander deutlich getrennte Unterthemen erkennbar. Im Zentrum der Debatte steht eine Gruppe von Schlüsselworten wie ‚Menschen', ‚Demokratie', ‚Europa', ‚fundamentalis*' und ‚menschenrecht*', welche auf eine zentrale Diskussion um Menschenrechte und den Islam in Europa hinweisen. Daneben sind zwei weitere Unterthemen auszumachen, die sich einerseits mit Religion, Toleranz und Glauben und auf der anderen Seite mit Terrorismus, Islamismus und Afghanistan befassen. Die Islamdebatte in der Schweiz war damit zwischen August 2009 und Mai 2010 vor allem von der Diskussion über den Islam als Religion vor dem Hintergrund der Islamisierung Europas geprägt. Die Krise im Nahen Osten wurde in dieser Zeit vornehmlich anhand von Anschlägen und den Taliban thematisiert.

Deutschland

In Deutschland zeichnet sich im beobachteten Zeitraum eine heterogene Debatte ab. Im Zentrum steht ein Unterthema, in dem vor allem auf Religion, Religiosität und Menschen in Deutschland Bezug genommen wird. Neben den Bezeichnungen verschiedener Religionen ist auch das Wort ‚Angst' in diesem Unterthema prominent, was auf Konflikte in der Debatte hinweist. Wie in der Schweiz sind auch in Deutschland zwei weitere Unterthemen zu finden, von denen sich eines mit Dialog und Toleranz beschäftigt, das andere mit dem Krieg im Nahen Osten. Im Fall von Deutschland sind im dialogorientierten Unterthema Worte wie ‚Türkisch' und ‚Identität' zentral, im Unterthema über den Nahen Osten sind dagegen ‚Afgahnistan', ‚Pakistan' und ‚Menschenrechte' vertreten. Die Analyse zeigt damit die Breite der Islam-Debatte in Deutschland auf, die im Spannungsfeld zwischen der Integration türkischstämmiger Einwanderer im Inland und dem militärischen Einsatz in Afghanistan stattfindet.

Frankreich

In Frankreich ist eine homogene Debatte über den Islam auszumachen, der durch ein zentrales Thema geprägt ist. Dieses zentrale Unterthema liegt zwischen ‚identité', ‚securité', ‚politique' und ‚concitoyen', was wie in Deutschland und der Schweiz auf eine Debatte über das Zusammenleben im Inland hinweist. Nah an diesem Unterthema liegen etwas weniger dicht Begriffe für Kultur und Integration. Abseits dieser zentralen Debatte ist ein Unterthema zur islamischen Religion zu erkennen. Anders als in Deutschland und der Schweiz fehlen Unterthemen zum Krieg im Nahen Osten und zum Terrorismus. Die Integration von Muslimen im Inland und die Spannungen in den Banlieues scheinen damit die Islam-Debatte in Frankreich im beobachteten Zeitraum dominiert zu haben.

Italien

In Italien ist anders als in den anderen Ländern kein zentrales Unterthema auszumachen. Die Debatte teilt sich in zwei Unterthemen, welche gleich weit von der Mitte des Scatterplots und damit vom Begriff ‚islamic*' entfernt sind. Das eine Unterthema befasst sich mit den Spannungen in Israel und enthält neben ‚Iran', ‚Israel' und den wichtigsten Städten auch Begriffe wie ‚terroris*' und ‚militar*'. Das zweite Unterthema enthält vor allem Begriffe um Einwohner, Zulassung und Einwanderung, was auf eine Debatte über die Zuwanderung von islamischen Menschen hinweist. Eine Debatte über die Religion, die Integration und den Dialog mit Muslimen ist dabei nicht zu erkennen. Ebenso fehlt eine Diskussion über den Krieg in Afghanistan. Die Islamdebatte in Italien ist folglich vom Konflikt in Israel und der Einwanderung beherrscht und weist kaum Parallelen zu den Debatten in den anderen Ländern auf.

Grossbritannien

Die Islamdebatte in Grossbritannien war im Untersuchungszeitraum von der Berichterstattung über Krieg und Konflikte im Nahen Osten geprägt. Sie ist in drei eng miteinander verbundene, aber dennoch getrennte Unterthemen gegliedert. Das prominenteste Unterthema enthält Worte wie ‚soldier*', ‚troop*', ‚iraq', ‚afghanistan' und ‚police', welche sich mit dem militärischen Einsatz in Afghanistan und Irak befassen. Eng damit verbunden ist auf der einen Seite ein Unterthema, in welchem über Terrorismus, Al-Qaida und Anschläge berichtet wird, auf der anderen Seite ein Unterthema mit Begriffen wie ‚law', ‚freedom', ‚population' und ‚peace'. Neben dieser zentralen Debatte über den militärischen Einsatz finden sich vereinzelte Begriffe, die sich dem Konflikt in Israel zuordnen lassen. Es fehlt jedoch jede innenpolitische Auseinandersetzung mit dem Islam. Weder Immigration noch Religionen oder Integration sind bei der Islamdebatte in Grossbritannien zentral.

Dänemark

Ähnlich wie in Grossbritannien ist auch in Dänemark hauptsächlich Auslandsberichterstattung über den Islam auszumachen. Das zentrale Unterthema bildet ein dichtes Cluster mit Worten wie ‚ledere' (Führer), ‚styre' (verwalten), ‚Obama' und ‚politisk' (politisch), welche auf Welt- und amerikanische Politik hinweisen. Daneben sind zwei eher heterogene Begriffswolken zu finden, die sich mit Spannungen in Israel und der Berichterstattung über den Islam zusammenfassen lassen. Fundamentalismus und Terrorismus sind mit wenigen, locker gestreuten Begriffen am Rand der Berichterstattung angezeigt, treten also nicht häufig gemeinsam auf. Eine Beschäftigung mit Islam im Inland oder mit Krieg ist in der SSA der Islamdebatte in Dänemark nicht auszumachen.

Fazit

In den untersuchten Ländern unterscheidet sich die Debatte über Islam und Islamisierung grundlegend, offenbar bedingt durch die Beteiligung an militärischen Einsätzen in islamischen Ländern und aktuelle Ereignisse im Inland. Gerade die unterschiedliche und teilweise fehlende Befassung mit dem Islam im Inland erschwert eine vergleichbare Befragung und Inhaltsanalyse über Islam und Islamisierung in den einzelnen Ländern. Gleichsam verhindern die mangelnden Gemeinsamkeiten die vergleichbare Befragung von Politikern, da je nach Land innenpolitische, aussenpolitische oder militärische Akteure zu unterschiedlichen Themen und Massnahmen befragt werden müssten.

Durch Term-Mapping konnten damit innert weniger Stunden Erkenntnisse über die Islamdebatten in verschiedenen Ländern gewonnen werden, welche zwar eine systematische Inhaltsanalyse nicht ersetzen können, für Vorentscheidungen bei einer internationalen Studie aber essentielle Informationen liefern. Durch das blinde Verfahren zur Auswahl der Schlüsselbegriffe waren zudem erst bei der Interpretation der Ergebnisse Übersetzungen notwendig.

Immigration

Das Thema der zweiten untersuchten Debatte ist Immigration. Auch hier wurde das Term-Mapping-Verfahren eingesetzt, um aus dem oben beschriebenen Korpus die Beziehungen von Schlüsselbegriffen zu ergründen. Die verwendeten Kernbegriffe waren Übersetzungen der Worte ‚Immigration' und ‚Einwanderung' und führten in allen Ländern zu ähnlichen Ergebnissen, jedoch waren durch die geringe Zahl an Artikeln zu diesem Thema die Ergebnisse in Deutschland und Dänemark nicht belastbar.

In den restlichen Ländern war die Debatte geprägt von Regulierung, unterschiedlichen Nationalitäten und Integration. Ebenso waren in allen Ländern rechte Parteien Teil der zentralen Debatte. Die grundlegenden Unterschiede bestanden im Rassismus, der in England ein sehr zentrales Thema ist, in Italien und der Schweiz dagegen am Rand der Debatte liegt, und in der Zentralität des Themas islamischer Einwanderer. Diese fällt vor allem in Frankreich und der Schweiz auf, wo auch das Kopftuch zu den zentralen Begriffen im Zusammenhang mit Immigration zählte.

Arbeitslosigkeit

Die dritte Debatte, welche zur Vorbereitung der internationalen Studie untersucht werden sollte, war Arbeitslosigkeit. Als Kernbegriffe wurden ‚arbeitslos' und ‚Arbeitslosigkeit' in allen fünf Sprachen verwendet. Für das Term-Mapping wurden auch hier zwischen 50 und 70 Begriffe in jedem Land gewählt.

Die Ergebnisse sprachen für starke Gemeinsamkeiten zwischen den Arbeitslosigkeitdebatten in den einzelnen Ländern. In allen Ländern konnten dieselben Unterthemen gefunden werden, gleichsam war die zentrale Debatte in allen Ländern durch ähnliche Begriffe gekennzeichnet. Diese Analyse eignet sich daher zur Betrachtung der Debatte anhand ihrer Unterthemen, wie im Folgenden aufgezeigt wird.

Wirtschaftskrise und Konjunktur

In allen sechs Ländern behandelte das zentrale Unterthema die internationale Wirtschafts- und Finanzkrise, welche seit Ende des Jahres 2008 auch die Wirtschaftslage europäischer Länder stark beeinflusst. Zu diesem zentralen Unterthema gehören jeweils auch Begriffe rund um die Arbeitslosenstatistik und den Arbeitsmarkt im Land. Die Unterschiede zwischen den Ländern beschränkten sich auf die Verteilung der Begriffe und einzelne landestypische Themenfoki – z.B. in der Schweiz Wachstum und Wettbewerbsfähigkeit. Nur in Dänemark war eine klare Trennung zwischen Arbeitsmarkt und Konjunktur zu sehen.

Arbeitslosenunterstützung und -betreuung

Die Unterstützung der Betroffenen wurde unterschiedlich stark gewichtet und unterschiedlich formuliert. Während in der Schweiz der finanzielle Aspekt im Zentrum stand, waren es in Deutschland die Vermittlung und Betreuung von Arbeitslosen. In Frankreich und Italien wurde die Unterstützung von Arbeitslosen nicht prominent thematisiert, in England hingegen gehörten die ‚Benefits' zum zentralen Unterthema der Wirtschafts- und Finanzkrise. Es ist damit anzunehmen, dass die Unterstützung von Arbeitslosen in den unterschiedlichen Ländern verschieden diskutiert und behandelt wird.

Besonderheiten

Landestypische Besonderheiten konnten am Rand der Debatten ausgemacht werden. So wurde in der Schweiz auch über Zuwanderung aus der EU und das Personenfreizügigkeitsabkommen gesprochen, in Italien wurde die ‚Exit Strategy' der amerikanischen Bundesbank thematisiert, in Frankreich wurden die Renten, die Jugendlichen und soziale Probleme angesprochen, und in England wurden die Pensionen und ‚Housing Benefits' erwähnt. Diese nationalen Eigenheiten sind für internationale Vergleiche von Bedeutung und sollten bei der Entwicklung von Instrumenten zur Befragung und Inhaltsanalyse bedacht werden, stehen jedoch der Vergleichbarkeit der Ergebnisse nicht im Weg.

Fazit

Mit der Arbeitslosigkeitsdebatte konnte eine latente Debatte identifiziert werden, welche in allen zu untersuchenden Ländern vergleichbar geführt wird. Die nationalen Besonderheiten, welche beim

Term-Mapping aufgefallen sind, können dabei bereits bei der Entwicklung der Instrumente wie dem Codebuch für die Inhaltsanalyse und dem Fragebogen für die Panel-Befragung berücksichtigt werden und geben dem Forscher bereits vor der Durchführung der Studie Hinweise auf zu erwartende Länderunterschiede.

Das Term-Mapping ersetzt dabei jedoch keine manuelle Inhaltsanalyse, da die Verwendung der einzelnen Begriffe und der Kontext, in dem sie genannt wurden, vom Verfahren nicht berücksichtigt werden. Es diente in diesem Fall ausschliesslich einem schnellen Überblick über die Berichterstattung in latenten Debatten, um Entscheidungen im Vorfeld einer Studie zu erleichtern.

Longitudinale Vergleiche

Neben der komparativen Anwendung des Term-Mapping ist auch eine longitudinale Anwendung denkbar und gegebenenfalls hilfreich, um die Entwicklung einer Debatte in einem Land oder einem Medium über die Zeit zu verfolgen. Dazu kann das Textkorpus in mehrere Zeitabschnitte unterteilt werden, in denen jeweils die Beziehung zwischen denselben Schlüsselworten analysiert wird.

Im vorliegenden Fall bot sich der zeitliche Verlauf der Islamdebatte in der Schweiz an, da in diesem Land im November 2009 über ein Bauverbot von Minaretten abgestimmt wurde, welches in der nationalen und internationalen Presse eine Diskussion über den Islam in der Schweiz und in Europa auslöste. Es war damit anzunehmen, dass sich die Debatte in der Schweiz in den Wochen vor und nach der Abstimmung verändert. Um diese Annahme zu testen, wurde der Untersuchungszeitraum in drei Abschnitte unterteilt.

1. Abschnitt: August bis Oktober 2009

Der erste untersuchte Zeitraum endet einen Monat vor der Abstimmung und war damit noch nicht von der Diskussion über das Minarett-Verbot dominiert. Das zentrale Unterthema enthielt entsprechend Begriffe wie ‚Ideologie', ‚Terrorismus' und ‚Araber' auf der einen Seite und ‚Werte', ‚Frauen' und ‚Ausland' auf der anderen Seite. Die Begriffe ‚Initiative', ‚Abstimmung' und ‚Tradition' kamen nur am Rand der Debatte vor und erschienen selten gemeinsam in einem Artikel.

2. Abschnitt: November 2009 bis Januar 2010

Der zweite untersuchte Zeitraum wurde im Umfeld der Abstimmung vom 29. November 2009 angelegt. Er enthält damit sowohl den Abstimmungskampf als auch die Reaktionen auf die Annahme des Minarett-Verbots. Erwartungsgemäss waren ‚Initiative', ‚Demokratie' und ‚Tradition', genauso wie ‚Integration' und ‚Unterdrückung' Teil der zentralen Diskussion zum Thema Islam in diesem Zeitraum. Begriffe, welche auf Terrorismus hinweisen, wurden hingegen an den Rand der Debatte gedrängt.

3. Abschnitt: Februar bis Mai 2010

Zu Beginn des dritten Zeitraums lag die Abstimmung bereits zwei Monate zurück und das Minarett-Verbot hatte nur noch einen entsprechend schwachen Einfluss auf die Medienberichterstattung dieser Monate. Die Begriffe ‚Initiative' und ‚Abstimmung' wurden erneut an den Rand der Debatte gedrängt, wobei die zentrale Diskussion vor allem ‚Terrorismus' auf der einen und ‚Werte' und ‚Debatte' auf der anderen Seite enthielt.

Die longitudinale Analyse mittels Term-Mapping hat somit die Tendenz bestätigt, welche durch ein einschneidendes Ereignis wie die Minarett-Initiative erwartet werden kann. Die Debatte rückte für eine kurze Zeit rund um das Ereignis von ihrem üblichen Fokus ab, um sich auf die Abstimmung zu konzentrieren. Danach wurde das Ereignis wieder aus der zentralen Diskussion verdrängt, welche sich erneut den anfänglich diskutierten Themen widmete.

Generelles Fazit

Das Term-Mapping-Verfahren erwies sich in der oberflächlichen Beschreibung von Debatten als wertvolles Hilfsmittel. Es ermöglicht eine Einschätzung der zentralen und peripheren Unterthemen von Debatten sowie deren Entwicklung über die Zeit. Die Ergebnisse sind jedoch rein qualitativ und es hängt vom Wissen des Forschers ab, die einzelnen Begriffe in einen sinnvollen Kontext zu setzen. Die Interpretation ist dadurch unter Umständen nicht intersubjektiv nachvollziehbar.

Der Vorteil des Verfahrens wird bei der Anwendung zur Vorbereitung einer international vergleichenden Studie jedoch deutlich: Der Inhalt der Berichterstattung zu drei Themen in sechs Ländern mit fünf unterschiedlichen Sprachen konnte in wenigen Tagen so weit analysiert werden, dass ein grober Überblick über Gemeinsamkeiten und Unterschiede zwischen den Ländern ersichtlich wird. Diese Geschwindigkeit und Effizienz des Term-Mapping macht es somit für die Eingrenzung des Forschungsgegenstandes und die Entwicklung von geeigneten Messinstrumenten zu einem wertvollen Hilfsmittel.

Diskussion und Implikationen

Bei der Vorbereitung von international vergleichenden Studien ist ein Überblick über die öffentlichen Debatten der Länder, in denen sie stattfinden, dringend nötig, sofern diese den Untersuchungsgegenstand berühren. Ebenso kann Bedarf nach einer schnellen und übersichtlichen Analyse verschiedener Debatten bestehen, um sich bei der Wahl eines Untersuchungsgegenstands auf international vergleichbar diskutierte Themen stützen zu können. Eine solche Erhebung ist jedoch

auch für automatische Inhaltsanalysen eine Herausforderung, da der Forscher objektiv und ohne Einfluss seines Alltagswissens vorgehen sollte und Sprachressourcen für alle beteiligten Länder bereitstellen muss.

Term-Mapping bietet in diesen Fällen eine Möglichkeit, innert weniger Tage einen Überblick über eine latente Debatte zu erhalten. Durch die Darstellung der Wahrscheinlichkeit des gemeinsamen Auftretens von Schlüsselworten mittels SSA wird die Debatte auf dem kleinstmöglichen Raum, einem zweidimensionalen Scatterplot, zusammengefasst. Die Auswertung gibt Aufschluss über die Homogenität der Debatte sowie die Anzahl, die Interaktion und den Inhalt der Unterthemen. Das Verfahren liefert somit eine leicht zu interpretierende Zusammenfassung von mehreren tausend Zeitungsartikeln oder Texten auf einem einzelnen Blatt.

Das Verfahren zeichnet sich jedoch durch zwei erhebliche Schwachstellen aus, die nicht vollständig beseitigt werden können. Zum einen ist die Interpretation der Scatterplots nicht intersubjektiv und lässt sich nicht durch konventionelle Kennwerte absichern, zum anderen wird der Kontext der Worte und damit ihre Einbettung in Argumente vernachlässigt, was zu falschen Deutungen einzelner Begriffscluster führen kann.

Während die Vernachlässigung des Kontexts bei diesem Verfahren bewusst in Kauf genommen wird, um eine intuitiv verständliche und möglichst einfache Darstellung einer komplexen Debatte zu erhalten, kann die Gruppierung der Begriffe durch andere Verfahren der multidimensionalen Skalierung bestätigt werden. Die auf gemeinsames Auftreten von Begriffen gestützte Korrelationsmatrix, welche als Zwischenergebnis vorliegt, lässt sich neben der SSA auch für eine Faktoranalyse (Crawley 2007) oder Clusteranalyse (Matthes/Kohring 2004; Miller/Andsager/Riechert 1998) verwenden.

Das Verfahren des Term-Mapping kann somit auch als Grundlage für weiterführende Analysen der Gruppierung von Begriffen in einer Debatte verwendet werden und ist nicht auf die oberflächliche Debattenanalyse beschränkt.

Literatur

Bloombaum, Milton: Doing Smallest Space Analysis. In: *Journal of Conflict Resolution* 14 (1970), Nr. 3, S. 409-416

Crawley, Catherine E.: Localized Debates of Argicultural Biotechnology in Community Newspapers: A Quantitative Content Analysis of Media Frames and Sources. In: *Science Communication* 28 (2007), Nr. 3, S. 314-346

DeLeeuw, Jan; Mair, Patrick: Multidimensional Scaling Using Majorization: SMACOF in R. In: *Journal of Statistical Software* 31 (2009), Nr. 3, S. 1-30

Früh, Werner: *Inhaltsanalyse: Theorie und Praxis*. Konstanz: UVK Verlagsgesellschaft, 2007.

Geis, Alfons: Konventionelle versus computergestützte Codierung offener Fragen: Ein Vergleich der Codierergebnisse. In: Wirth, W.; Lauf, E. (Hrsg.): *Inhaltsanalyse: Perspektiven, Probleme, Potentiale*. Köln: Halem (2001), S. 318-337

Guttmann, Louis: A general nonmetric technique for finding the smallest coordinate space for a configuration of points. In: *Psychometrika* 33 (1968), Nr. 4, S. 469-506

Krippendorff, Klaus: *Content Analysis: An Introduction to its Methology*. Thousand Oaks: Sage, 2004.

Kuckartz, Udo: *Einführung in die computergestützte Analyse qualitativer Daten*. Wiesbaden: VS Verlag für Sozialwissenschaften, 2007.

Matthes, Jörg; Kohring, Matthias: Die empirische Erfassung von Medien-Frames. In: *M&K* 52 (2004), Nr. 1, S. 56-78

Miller, Mark; Riechert, Bonnie P.: Frame Mapping: A Quantitative Method for Investigating Issues in the Public Sphere. In: West, D. (Hrsg.): *Theory, Method, and Practice in Computer Content Analysis*. London: Ablex, 2001, S. 61-75

Miller, Mark M.; Andsager, Julie L.; Riechert, Bonnie P.: Framing the Candidates in Presidential Primaries: Issues and Images in Press Releases and News Coverage. In: *Journalism & Mass Communication Quarterly* 75 (1998), Nr. 2, S. 312-324.

NCCR: *Module 4: Changing processes and strategies of political participation and representation: comparing public debates*. URL http://www.nccr-democracy.uzh.ch/research/module-4. – Aktualisierungsdatum: Februar 2011. – mailto: w.wirth@ipmz.uzh.ch. – NCCRdemocracy.

Abbildungen

$$P_{(w1|w2)} = \frac{N_{gemeinsam}}{N_{w2}} \quad (1)$$

$$P_{w12} = P_{(w1|w2)} \cdot P_{(w2|w1)} = \frac{N^2_{gemeinsam}}{N_{w1} \cdot N_{w2}} \quad (2)$$

$$D_{w12} = \begin{cases} P_{w12} > 0: \ -\log\left(\frac{N_{w1} \cdot N_{w2}}{N^2_{gemeinsam}}\right) \\ P_{w12} = 0: \ -\log\left(\frac{N_{w1} \cdot N_{w2}}{0.25}\right) \end{cases} \quad (3)$$

Abbildung 1

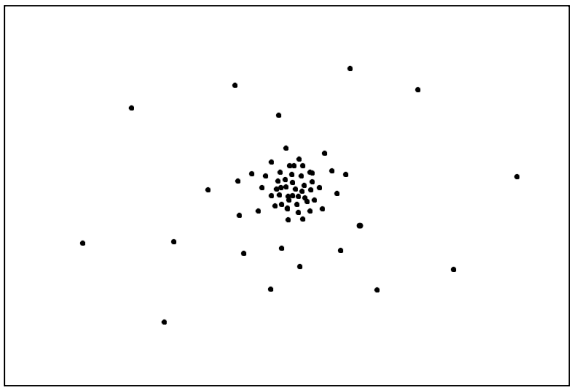


Abbildung 2a

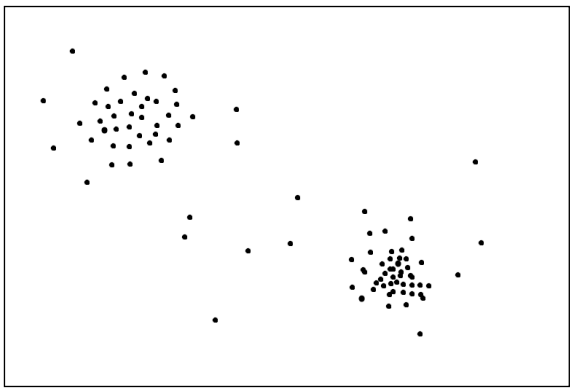


Abbildung 2b

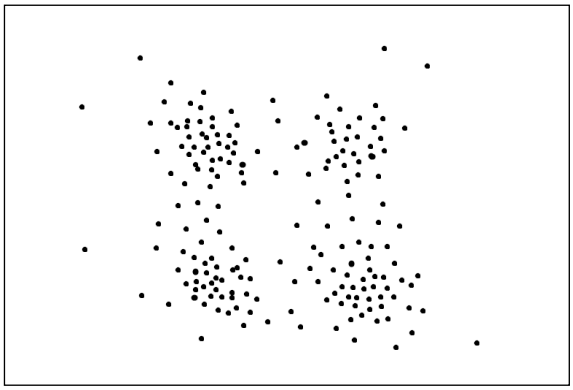


Abbildung 2c

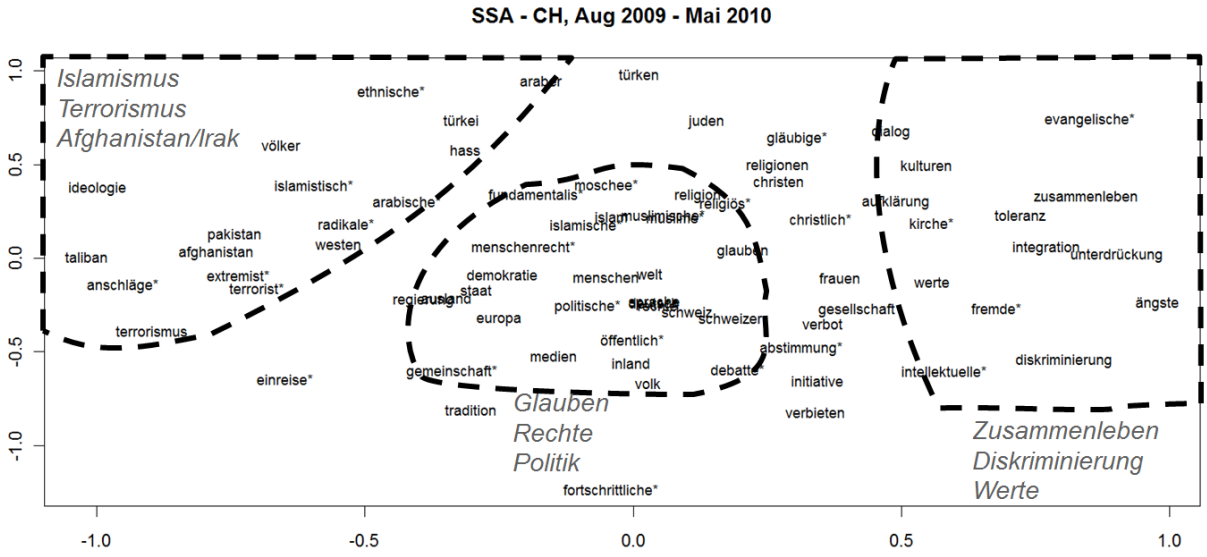


Abbildung 3a

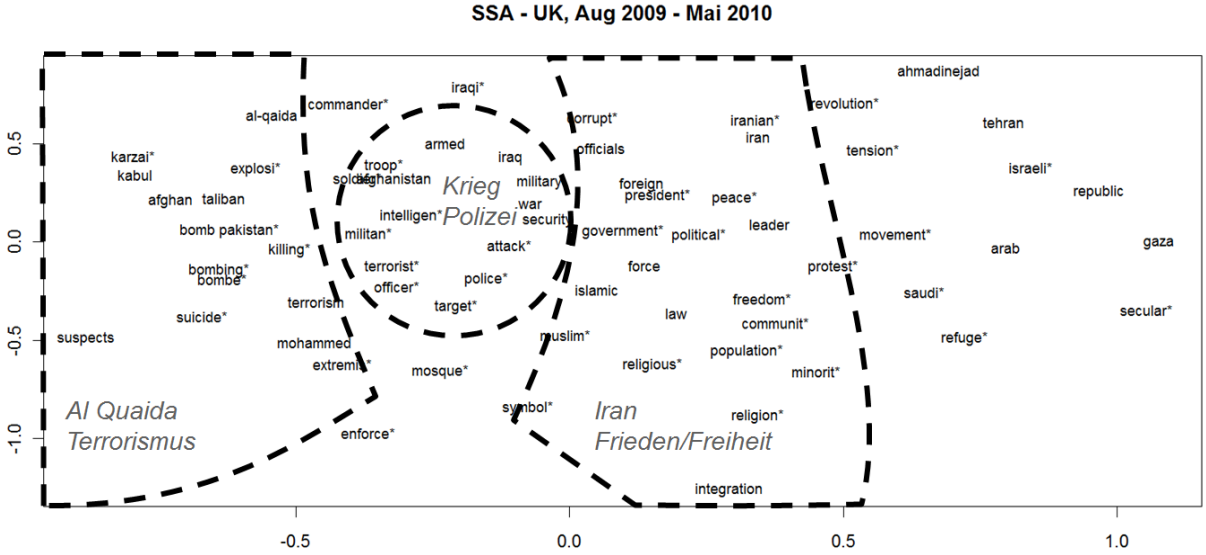


Abbildung 3b

Anhang A2: Best of Both Worlds

Veröffentlicht als:

Wettstein, M. (2014). "Best of Both Worlds": Die halbautomatische Inhaltsanalyse. In K. Sommer, M. Wettstein, W. Wirth, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse* (pp. 16–39). Köln: von Halem.

«Best of both worlds»: Die halbautomatische Inhaltsanalyse

[Bitte nicht aus diesem Manuskript zitieren. Manuskript zur Publikation: Wettstein, M. (2014). "Best of Both Worlds": Die halbautomatische Inhaltsanalyse. In K. Sommer, M. Wettstein, W. Wirth, & J. Matthes (Eds.), Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse (pp. 16–39). Köln: von Halem.]

Die schnell wachsende Verfügbarkeit von elektronischen Texten, seien dies digitale Ausgaben von Zeitungen, Formen nutzergenerierter Inhalte in Online-Medien oder Textnachrichten in sozialen Netzwerkdiensten, führten in den letzten Jahren zu intensiven Bemühungen, Verfahren der computergestützten Inhaltsanalyse für diverse Fragestellungen der Kommunikationswissenschaft dienstbar zu machen. Diese Verfahren, bei welchen Texte nicht mehr manuell durch geschulte Codierer, sondern automatisch durch spezialisierte Programme erfasst und ausgewertet werden, haben den unstrittigen Vorteil, schnell und effizient große Mengen an Text bearbeiten zu können. Gerade bei der Analyse von Big Data, also großer Mengen von Text, welche über Zeitungsdatenbanken oder über die Beobachtung von Online-Kommunikation gewonnen werden, ist dies von Vorteil.

Dennoch bestehen gerade in der Kommunikationswissenschaft noch gewisse Vorbehalte gegenüber vollautomatischen Analyseverfahren. Nach Jahrzehnten der methodologischen Auseinandersetzung mit manuellen Inhaltsanalyseverfahren wurde in diesem Wissenschaftszweig ein Bewusstsein für die Rolle und die Aufgabe menschlicher Codierer geschaffen, welches die Akzeptanz von Computerprogrammen als Ersatz erschwert. Der Codierer, so schreibt Früh (2009), muss einen Text verstehen und abstrahieren, Codiereinheiten verlässlich identifizieren und sie anschließend anhand eines Codebuches klassifizieren (S.134f). Dazu benötigt er Fähigkeiten wie Sprachkompetenz, Abstraktionsvermögen und Mustererkennung und muss über ein gewisses Weltwissen verfügen, welches es ihm ermöglicht, auch implizite und metaphorische Bestandteile des Textes korrekt zu interpretieren. Selbst die aktuellsten Computerprogramme sind nicht in der Lage, diese Aufgaben vollumfänglich zu übernehmen und dabei die Anforderungen an die Reliabilität und Validität, welche in der Kommunikationswissenschaft an Codierer gestellt werden, zu erfüllen (Conway, 2006, Wüest, Clematide, Bünzli, Laupper, & Frey, 2011).

Trotz dieses begründeten Einwands gegen die Substitution manueller durch automatische Inhaltsanalyse feierte diese in den letzten Jahren zahlreiche Erfolge und kommt immer öfter auch in hochrangig publizierten Inhaltsanalysen zum Einsatz (e.g.: Colleoni, Rozza, & Arvidsson 2014; Sjøvaag, Moe, & Stavelin 2012; van Atteveldt, Kleinnijenhuis, & Ruigrok 2008). Ohne Texte verstehen oder in einen größeren Kontext setzen zu können, erlauben es ausgefeilte Algorithmen, bestimmte Informationen aus Texten zu lesen, sie aufgrund von einzelnen Begriffen oder Begriffskonstellationen korrekt zu klassifizieren oder sogar Beziehungen von Objekten in Texten zu bestimmen. Einzelne

Fragestellungen lassen sich dadurch mit Hilfe automatischer Verfahren durchaus beantworten. Dabei gilt es jedoch zu bedenken, dass insbesondere komplexe Verfahren zur Extraktion von Bedeutung aus Texten äußerst voraussetzungsreich sind und meist sowohl Programmierkenntnisse als auch einen erheblichen Zeitaufwand zur Vorbereitung einer Inhaltsanalyse nötig machen (Scharkow, 2012).

In diesem Kapitel soll ein Verfahren vorgestellt und in der inhaltsanalytischen Praxis erprobt werden, welches die Vorteile der manuellen Inhaltsanalyse (insbesondere die Interpretationsfähigkeit, das Weltwissen und die Validität) mit den Vorteilen der computergestützten Inhaltsanalyse (insbesondere die Effizienz und Gründlichkeit) kombinieren. Die daraus resultierende Methode der halbautomatischen Inhaltsanalyse ist für die Codierer mit erheblich weniger Aufwand verbunden, da sie durch computergestützte Verfahren in ihrer Arbeit unterstützt werden. Für den Forscher bedeutet dies, mehr Texte mit demselben Zeitaufwand für Codierer bearbeiten zu können.

Konzept

Die Idee manuelle Inhaltsanalysen durch automatische Verfahren zu ergänzen ist nicht neu (vgl. Stuckardt 2000). Meist wird bei diesen Formen der computerunterstützten Inhaltsanalyse ein bestimmter Schritt des Analyseprozesses durch ein geeignetes Computerprogramm durchgeführt. Am weitesten verbreitet ist dabei wohl die Suche nach relevanten Texten in umfangreichen Datenbanken mittels Schlüsselworten oder trainierten Klassifikationsalgorithmen (siehe Scharkow, 2012), welche die manuelle Suche nach geeigneten Texten aus Zeitungen ablöst. Des Weiteren werden automatische Verfahren zur vor- oder Nachbereitung der Erhebung genutzt. So kann eine statistische Textanalyse im Vorfeld von Inhaltsanalysen helfen, den untersuchten Themenbereich abzuschätzen und einzugrenzen (vgl. Wettstein, 2012). Ebenso kann die manuelle Inhaltsanalyse sich auf das Auffinden semantischer Elemente beschränken, um im Anschluss an die Erhebung statistischer Verfahren Muster zu identifizieren und zu interpretieren (Matthes & Kohring, 2008). Ein anderer Ansatz besteht darin, parallel zur manuellen Inhaltsanalyse, welche sich auf die inhaltliche Bedeutung von Texten beschränkt, eine automatische Analyse durchzuführen, welche computerlesbare Eigenschaften derselben Texte (z.B: Länge der Texte oder das Vorkommen bestimmter Begriffe) erhebt. Im Anschluss an die beiden Erhebungen können die Daten der einzelnen Analysen zu einem Datensatz kombiniert werden.

Die Idee der halbautomatischen Inhaltsanalyse verfolgt jedoch einen ganz anderen Ansatz. Anstatt computergestützte Verfahren vor, nach oder parallel zur manuellen Inhaltsanalyse durchzuführen, werden bei diesem Verfahren die manuelle und automatische Inhaltsanalyse gleichzeitig durchgeführt und beeinflussen einander. Konkret soll dabei der Codierer die Daten direkt am Computer über eine Eingabemaske erfassen und dabei von computergestützten Verfahren im Hintergrund unterstützt und entlastet werden. Die Art der im Hintergrund laufenden Verfahren ist

dabei je nach Fragestellung der Inhaltsanalyse und die Gestaltung der Kategorien unterschiedlich. Falls ein Verfahren gewählt wird, welches aus Eingaben des Codierers lernen kann, so kann es während der Inhaltsanalyse kontinuierlich trainiert und verfeinert werden.

Umsetzung

Die halbautomatische Inhaltsanalyse, wie sie in diesem Kapitel beschrieben wird, erfordert zwei miteinander kombinierbare Komponenten. Zum einen ist dies eine Eingabemaske, über welche der Codierer die Daten der Inhaltsanalyse direkt eingeben kann, zum anderen ein Programm zur automatischen Inhaltsanalyse von Texten. Um diese beiden Komponenten zu verbinden muss es möglich sein, den aktuell bearbeiteten Text von der Eingabemaske an die automatische Inhaltsanalyse zu übergeben. Gleichzeitig muss es möglich sein, die Ergebnisse der automatischen Inhaltsanalyse während der Dateneingabe in die Eingabemaske einzubinden, dass der Codierer bei seinen Entscheidungen entlastet wird.

Im Folgenden werden zunächst die Anforderungen an die Eingabemaske und geeignete Verfahren der automatischen Inhaltsanalyse getrennt behandelt, um im Anschluss die Kombinationsmöglichkeiten in der halbautomatischen Inhaltsanalyse zu diskutieren.

Eingabemaske

Aktuell sind mehrere Programme verfügbar, mit welchen sowohl qualitative als auch quantitative Inhaltsanalysen direkt am Computer durchgeführt werden können. Grundsätzlich bestehen diese Programme meist aus einer Eingabemaske oder einem Formular, über welches der Codierer seine Entscheidungen direkt eintragen kann, und einer Möglichkeit, den Text oder den audiovisuellen Beitrag am Bildschirm anzuzeigen und zu durchsuchen. Die Programme sind dabei entweder lokal installiert, was einen regelmäßigen Datenaustausch mit der Projektleitung erfordert, oder sie sind als Online-Formular konzipiert und speichern die Daten zentral auf einem Server.

Für eine Verwendung zur halbautomatischen Inhaltsanalyse ist es unerheblich, in welcher Form die Eingabemaske vorliegt, so lange zwei Grundvoraussetzungen gegeben sind: Erstens muss sich das Formular, auf welchem der Codierer seine Daten eingibt, durch ein im Hintergrund laufendes Programm verändern lassen. Dies betrifft sowohl die Vorauswahl bestimmter Ausprägungen als auch die Sortierung und Darstellung von Ausprägungen in Fällen, in welchen der Codierer aus einer Liste von Ausprägungen eine Wahl treffen muss. Zweitens müssen die Inhalte, welche der Codierer aktuell bearbeitet, laufend an dieses Programm gesendet werden können.

Da es bei der Arbeit mit kommerziellen Programmen zur Inhaltsanalyse schwierig ist, ein Programm für automatische Inhaltsanalyse so einzubinden, dass es sowohl Informationen über die aktuelle Codierung erhält als auch die Eingabefelder beeinflussen kann, empfiehlt sich die Arbeit mit

einem selbst aufgesetzten Online-Formular oder einem leicht zu bearbeitenden Open Source Programm, welches die Einbindung fremder Programme zulässt. In der konkreten Anwendung, welche in diesem Kapitel vorgestellt wird, wurde die das Open Source Programm Angrist (Wettstein, 2013) verwendet, welches sich als Python-Skript leicht mit Programmen zur automatischen Inhaltsanalyse verbinden lässt und eine dynamische Anpassung der Oberfläche während der Codierung erlaubt.

Automatische Inhaltsanalyse

Zur automatischen Analyse von Texten wurde in den vergangenen Jahrzehnten eine Vielzahl von Verfahren und Programmen entwickelt, welche sich grob in drei grundlegend unterschiedliche Ansätze einteilen lassen (vgl. Skalski, 2002; Krippendorff, 2004: 281ff; Young & Soroka, 2012). Der einfachste Ansatz sind naive Text-Mining Verfahren. Diese extrahieren Informationen aus einem Text, indem sie nach bestimmten Zeichen, Worten oder Mustern suchen. In diese Kategorie fallen alle Verfahren, welche mit der Volltext-Suche nach Schlüsselworten arbeiten oder Informationen aus Kopfzeilen oder standardisierten Textelementen ziehen. Ein weiterer Ansatz sind statistische Verfahren, welche Informationen aus einer Vielzahl von Texten extrahieren, indem sie das Auftreten von einzelnen Begriffen und deren gemeinsames Vorkommen statistisch auswerten. In diese Kategorie fallen sowohl explorative Verfahren zur vereinfachten Darstellung großer Textmengen (Landmann & Züll, 2004) als auch trainierte Verfahren zur Text-Klassifikation (Scharkow, 2012). Als dritten Ansatz kann die computerlinguistische Herangehensweise gesehen werden, einen Text nicht nur über einzelne Begriffe und deren Auftreten, sondern über die Grammatik und Semantik zu entschlüsseln und Inhalte in eine formale Sprache zu übersetzen (Roberts, 1989).

Alle drei Ansätze lassen sich theoretisch für eine halbautomatische Inhaltsanalyse verwenden und haben je nach verwendetem Kategorie-System bestimmte Vor- und Nachteile. Im Folgenden werden die konkreten Einsatzmöglichkeiten für die drei Ansätze vorgestellt und diskutiert.

Text-Mining

Bei Text-Mining-Ansätzen werden Texte als Datenmenge verstanden, welche sich nach bestimmten Informationen wie dem Auftreten von Schlagworten oder der Anzahl Begriffe durchsuchen lässt. Diese Verfahren sind daher besonders für die Erfassung von formalen Kategorien geeignet. Text-Mining-Verfahren sind leicht und meist ohne tiefgehende Programmierkenntnisse umzusetzen. Sie basieren auf einfachen Regeln, welche einen Text nach bestimmten Zeichen oder Zeichenketten durchsuchen.

Ein Beispiel für eine solche Regel ist die Suche und Zählung von Leerzeichen und Zeilenumbrüche in einem Text. Die Summe dieser beiden Zahlen ergibt grob die Anzahl der Worte eines Textes, welche damit automatisch erhoben werden kann. Gerade diese Eigenschaft von Texten ist in

manuellen Inhaltsanalysen mit Aufwand für den Codierer verbunden, der die Worte zählen und hochrechnen muss. Ein anderes Beispiel ist die Suche nach standardisierten Textelementen, welche alle Texte in der Inhaltsanalyse enthalten. Bei HTML-Texten sind dies Tags (z.B: <img>, der auf ein Bild hinweist oder <h1>...</h1> welche den Titel einschliessen), bei standardisierten Einträgen in einer Textdatenbank können es auch andere Textelemente sein (z.B: 'SECTION:' welches in Lexis-Nexis die Rubrik des Textes einleitet). Wird nach diesen Elementen gesucht, können bestimmte Informationen vollautomatisch erhoben werden.

Gerade für die halbautomatische Inhaltsanalyse bietet sich eine weitere Einsatzmöglichkeit für Text-Mining Verfahren. Wird beispielsweise das Auftreten einer langen Liste von Akteuren erhoben, so kann diese Liste vor Beginn der Codierung um die unmöglich vorhandenen Akteure reduziert werden. Kommt ein Name auf der möglicherweise sehr umfangreichen Liste aller Akteue im Text nicht vor (weder Vor- noch Nachname noch die Funktion), so ist es unwahrscheinlich, dass dieser Akteur im aktuellen Text codiert wird. Die Arbeit des Codierers kann erheblich vereinfacht werden, wenn nur jene Akteure auf der Liste verbleiben, deren Namen (oder Bestandteile davon) im Artikel vorkommen. Dabei verbleiben zwar möglicherweise fünf Akteure namens 'Müller' auf der Liste, obschon nur einer gemeint ist, dem Codierer fällt die Auswahl aber leichter, da die Liste insgesamt kürzer und übersichtlicher ist. Ähnliche Einschränkungen lassen sich auch bei anderen Listen mit Ausprägungen vornehmen, deren Vorkommen an bestimmte Worte gebunden ist.

Bei Text-Mining-Verfahren ist allgemein zu beachten, dass das Vorkommen einzelner Zeichenketten kaum Auskunft über den Inhalt oder die Bedeutung von Texten geben kann. Die Bedeutung von Sätzen oder gar ganzen Texten lässt sich nur schwer aus einzelnen Worten ableiten, wodurch diese Verfahren sich eher für formale Kategorien als für inhaltliche Codierungen eignen. Für die obengenannten Beispiele sind sie jedoch äußerst sinnvoll und lassen sich leicht implementieren.

Bag of Words Ansatz

Beim Bag-of-Words (BoW) Ansatz wird ein Text als eine unsortierte Ansammlung von Worten verstanden, aus deren (gemeinsamem) Auftreten Informationen über den Inhalt des Textes gewonnen werden können. Für diesen Ansatz ist die genaue Verwendung, der Kontext oder die Reihenfolge der betrachteten Worte irrelevant und werden bei der Analyse nicht berücksichtigt. Trotz dieser Einschränkung hat sich eine Vielzahl von hilfreichen Verfahren zur Textanalyse entwickelt, welche auf diesem Ansatz beruhen. So gibt es zum einen nicht-statistische Verfahren, bei welchen das Auftreten von Begriffen deduktiv mit Hilfe von vordefinierten Wortlisten analysiert wird, um die Tonalität (Hart; Stewart, Pitts, & Osborne, 2011; Young & Soroka, 2012) oder das Thema (Conway, 2006) eines Textes zu erkennen. Zum anderen kann mit statistischen BoW-Verfahren (vgl. Manning & Schütze, 1999) das Auftreten von Begriffen induktiv statistisch ausgewertet werden, um

das Framing unterschiedlicher Quellen (Danowski, 1993; Miller & Riechert, 2001) oder die Unterthemen von Debatten (Crawley, 2007; Kim, 2011) zu ermitteln.

Mit der trainierten Textklassifikation hat sich ein BoW-Verfahren herausgebildet, welches für die halbautomatische Inhaltsanalyse von großer Bedeutung ist. Bei diesem Verfahren werden Texte in einem ersten Schritt manuell kategorisiert. In einem zweiten Schritt wird das Vorkommen einzelner Begriffe in den kategorisierten Texten daraufhin getestet, ob sie in einer Kategorie besonders häufig vorkommen. Liegt eine kleine Menge korrekt klassifizierter Texte vor, können aufgrund der Informationen über Worthäufigkeiten in den einzelnen Kategorien unbekannte Texte automatisch kategorisiert werden. Durch eine laufende manuelle Kontrolle der automatischen Klassifikation kann das Programm laufend lernen und validere Entscheidungen treffen (für eine ausführliche Beschreibung siehe: Scharkow, 2012). Diese Verfahren werden aktuell zur Klassifikation von Texten nach Themenbereich oder in Verbindung mit Spam-Filtern verwendet, um große Mengen von Texten einer Kategorie zuzuordnen.

Gerade die Eigenschaft trainierter statistischer Verfahren, anhand menschlicher Entscheidungen zu lernen und sich zu verbessern entspricht dem Konzept der halbautomatischen Inhaltsanalyse, welches aus der Wechselwirkung zwischen Mensch und Maschine einen Nutzen ziehen möchte. Ähnlich wie bei der Klassifikation eines Textes als Spam oder als relevanter Text können diese Verfahren auch eingesetzt werden, um einzelne Kategorien von Inhaltsanalysen automatisch zu codieren. So kann für eine Vielzahl von Kategorien (z.B: Themenfeld, Geographische Reichweite, Skandalisierung) ein solches Verfahren eingesetzt werden, um die plausibelste Ausprägung im Vorfeld der Codierung zu ermitteln und dem Codierer als Vorschlag zu unterbreiten. Der Codierer muss die Vorgaben nur noch kontrollieren und nicht mehr manuell eintragen, was Zeit und kognitiven Aufwand spart sowie das Risiko für heuristische Codierentscheidungen (aufgrund zu großer und unübersichtlicher Listen) reduziert (vgl. Wirth, 2001).

Zwei Eigenheiten dieser Verfahren sollten jedoch bei ihrem Einsatz bei Inhaltsanalysen auf jeden Fall beachtet werden. Zum einen sind auch gut trainierte statistische Verfahren nicht perfekt und erreichen nur bei sehr einfachen Kategorien je die Validität, welche bei menschlichen Codierern vorausgesetzt wird. Sie können zwar darauf trainiert werden, um einen bestimmten Fehler möglichst selten zu begehen (z.B. eine wichtige Nachricht fälschlicherweise als Spam zu kennzeichnen), aber dies führt meist zu vermehrten Fehlern der entgegengesetzten Richtung (also einige Spam-Nachrichten werden nicht als solche erkannt). Zum anderen muss bedacht werden, dass diese Verfahren alleine aufgrund des Vorkommens einzelner Begriffe entscheiden, wie ein Text zu klassifizieren ist. Dies funktioniert besonders gut bei der Codierung von Politikfeldern und anderen Kategorien, für welche auch der menschliche Codierer einzelne Begriffe als Entscheidungshilfe

heranzieht. Da die Verfahren jedoch den Zusammenhang der Begriffe, die Satzstellung, Negationen und den Kontext völlig außer Acht lassen, ist es nicht möglich, damit den Sprachstil, die Emotionalisierung, Sarkasmus oder andere stilistische Feinheiten zu erkennen. Hier braucht es die Sprachkompetenz eines menschlichen Lesers, der nicht nur die Begriffe, sondern auch die Formulierung erkennen kann.

Die fehlende Berücksichtigung von Kontext und Satzbau ist jedoch nicht nur negativ zu sehen. Da für den Einsatz trainierte Klassifikation die Grammatik und der Kontext außer Acht gelassen werden, ist dieses Verfahren auf unterschiedliche Sprachen und unterschiedliche Zusammenhänge anwendbar. Dieser Vorteil und der geringe Aufwand bei der Vorbereitung dieser Verfahren haben sicher zu ihrer aktuell sehr starken Verbreitung in der Praxis (vgl. Young & Soroka, 2012) beigetragen.

Semantisch/Grammatikalische Verfahren

Der dritte Ansatz, welcher vor allem in der Computerlinguistik angewandt wird, sieht einen Text als eine Ansammlung konkreter Aussagen einer natürlichen Sprache, welche klaren grammatikalischen und semantischen Regeln folgt und in eine formale Sprache übersetzt werden kann. Die formale Repräsentation eines Satzes kann dann verwendet werden, um die Aussage von Texten nachzuvollziehen (Roberts, 1989). Verfahren, welche auf diesem Ansatz beruhen, verwenden ausführliche Regeldatenbanken und Vergleichskorpora, um den Gehalt von Sätzen genau bestimmen und auswerten zu können. Sie sind damit stark sprachabhängig, können aber bis zu einem gewissen Grad Argumentationsstrukturen und rhetorische Feinheiten erfassen.

So wurden in der Vergangenheit die Darstellung einzelner Akteure in Zeitungsberichten (van Atteveldt, Kleinnijenhuis, & Ruigrok, 2008), zentrale Ergebnisse wissenschaftlicher Texte (Schneider, Clematide, & Rinaldi, 2011) oder die Beziehung zwischen politischen Akteuren und Themenfeldern (Wüest et al., 2011) automatisch aus Texten extrahiert. Die Komplexität menschlicher Sprache, insbesondere in journalistischen Texten, welche regen Gebrauch von Metaphern, Ironie oder Rückbezügen auf vorangehende Sätze und Absätze machen, stellt jedoch die Computerlinguistik vor große Herausforderungen (Wüest et al., 2011). Zudem erfordern diese Verfahren intensive Vorarbeit bei der Definition von Lexika und grammatischen Regeln, welche für jede Sprache neu geleistet werden muss. Die Kosten für diese Verfahren sind dadurch entsprechend hoch (Scharkow, 2012).

Während die Vorbereitung und die Kosten bei einem Projekt durchaus in Kauf genommen werden können, stellt die mangelnde Validität automatischer Erhebungen sprachlich anspruchsvoller Texte ein Problem für computergestützte Analysen dar. Bei der halbautomatischen Inhaltsanalyse sind die zu erwartenden Probleme jedoch deutlich geringer, da hier der menschliche Codierer als letzte Instanz aller Entscheidungen fungiert. So können mögliche Fehler korrigiert und die Validität der

Codierung gesichert werden. Gerade in Inhaltsanalysen, in denen Beziehungen zwischen Objekten im Text erkannt werden sollen, können diese Verfahren sehr hilfreich sein. Sie können nicht nur Objekte und deren Beziehungen ermitteln, sondern den Codierer gleichzeitig auf die entsprechende Textstelle hinweisen und ihm damit die Codierarbeit erleichtern.

Kombination von Vorhersage und Eingabe

Für eine halbautomatische Inhaltsanalyse ist es nun notwendig, die computergestützte Inhaltsanalyse im Hintergrund mit der manuellen Eingabe zu verbinden. Dabei ist eine Datenübermittlung in beide Richtungen, also von der Eingabemaske an die automatische Inhaltsanalyse und umgekehrt, erforderlich.

Zunächst muss der aktuell bearbeitete Text oder die konkrete Textstelle an die automatische Inhaltsanalyse übergeben werden. In Fällen, in welchen der gesamte Text analysiert wird, muss zumindest eine elektronische Version des Textes für die automatische Inhaltsanalyse zu Verfügung stehen. Falls jedoch eine Inhaltsanalyse mehrere Analyseeinheiten pro Text unterscheidet (z.B.: Abschnitte oder Aussagen), sollte jeweils nur der Textabschnitt automatisch analysiert werden, welchen der Codierer aktuell bearbeitet. In diesem Fall sollte dem Codierer die Möglichkeit gegeben werden, die aktuell bearbeitete Textstelle zu markieren, so dass das Programm nur den markierten Text zu beachten braucht.

Während die Übermittlung von Text an ein Programm in den meisten Fällen sehr leicht zu realisieren ist, stellt sich der gegenläufige Datenaustausch, also die Einbindung von Ergebnissen der automatischen Inhaltsanalyse, komplexer dar. Hier muss ein Weg gefunden werden, den Codierer im Fall eindeutiger Ergebnisse zu entlasten, im Fall ambivalenter Ergebnisse zu beraten und in unklaren Situationen nicht durch potentiell falsche Vorschläge zu verwirren. Im Folgenden werden daher für vier verschiedene Situationen Möglichkeiten diskutiert, welche sich bei der Kombination der Ergebnisse einer automatischen Inhaltsanalyse mit einer Eingabemaske anbieten.

Situation 1: Das Programm liefert die korrekte Lösung

In einzelnen Fällen ist es möglich, durch Textmining eine Kategorie automatisch zu codieren. Oben wurde in diesem Zusammenhang bereits die Länge des Textes als Beispiel angegeben. Ebenso lassen sich aber auch der Titel eines Artikels oder Felder in einem standardisierten Format durchaus automatisch auslesen. Falls der Algorithmus für diese Kategorien eine sinnvolle Lösung gefunden hat (z.B. falls das ausgelesene Datum die Form XX.XX.XXXX hat), muss der Codierer nicht mit dieser Kategorie aufgehalten werden. Die Verknüpfung zwischen automatischer Inhaltsanalyse und Eingabemaske sollte für diese Situation also erlauben, dass eine Kategorie automatisch bestimmt wird, ohne dass der Codierer das Ergebnis prüft.

Nur in sehr klaren Situationen ist es jedoch sinnvoll, eine Kategorie direkt durch das Programm bestimmen zu lassen, das Feld in der Eingabemaske automatisch auszufüllen und die Eingabe zu bestätigen. Auch hier ist es jedoch ratsam, das Programm für mögliche Fehlerquellen zu sensibilisieren und Regeln aufzustellen, wie eine plausible Codierentscheidung aussehen müsste. So kann zum Beispiel die Länge eines Titels zwischen zwei und zehn Worten betragen. Ist er länger oder kürzer, kann der Codierer gebeten werden, die Eingabe manuell zu bestätigen. Gleiches gilt für Daten im falschen Format, für unbekannte Autoren und Rubriken oder falls die Anzahl Worte und Zeichen in einem Text unrealistisch hoch oder tief ist.

Situation 2: Das Programm liefert eine wahrscheinlich korrekte Lösung

Bei der automatischen Codierung einer Kategorie durch statistische BoW-Verfahren kann für jede mögliche Lösung eine bestimmte Wahrscheinlichkeit angegeben werden. So gibt es Ausprägungen, welche für einen bestimmten Text eine hohe oder tiefe Wahrscheinlichkeit haben, also plausibel oder weniger plausibel sind. Keine der Ausprägungen kann jemals durch ein solches Verfahren sicher angenommen oder ausgeschlossen werden. Für diese Verfahren bietet es sich also an, die automatische Codierung immer durch den Codierer prüfen und manuell bestätigen zu lassen. Bei der Verknüpfung zwischen automatischer Inhaltsanalyse und Eingabemaske sollte es hier möglich sein, automatisch eine plausible Auswahl für eine Kategorie zu treffen und diese durch den Codierer manuell bestätigen zu lassen.

Konkrete Vorschläge für eine Kategorie sollten jedoch nur dann unterbreitet werden, wenn eine Ausprägung sich als sehr wahrscheinlich erweist. Falls die Wahrscheinlichkeit der plausibelsten Ausprägung nur geringfügig höher ist als jene einer Alternative ist es ratsam, den Codierer um eine manuelle Eingabe zu bitten, ohne ihm einen Vorschlag zu unterbreiten. Je nach Kategorie und Leistung des Programms kann hier eine Grenze bestimmt werden, ab welcher das Programm keine Vorschläge mehr macht.

Situation 3: Das Programm liefert mehrere Lösungen für Auswahllisten

Die automatische Bestimmung der wahrscheinlichsten Entscheidung ist bei multinominalen Kategorien mit mehr als fünf Ausprägungen möglicherweise problematisch. Hier besteht immer die Möglichkeit, dass mehrere Ausprägungen eine ähnlich hohe Wahrscheinlichkeit aufweisen und daher gleichzeitig in Betracht kommen. Für diese Situation sollte es möglich sein, besonders plausible Ausprägungen für den Codierer durch eine typographische Hervorhebung oder einen prominenten Platz auf der Auswahlliste gut sichtbar zu machen. Für die Verknüpfung automatischer Verfahren mit der Eingabemaske ist es hier also sinnvoll, die Ergebnisse der automatischen Inhaltsanalyse in die Darstellung von Listen einfließen zu lassen und die Reihenfolge oder Schrift der Einträge zu verändern.

Nur selten ist es möglich, einzelne Ausprägungen auf langen Listen für einen gegebenen Text oder eine Textstelle mit Sicherheit auszuschließen. In diesem Fall kann der Codierer bei der Auswahl dadurch unterstützt werden, dass die Auswahlliste auf die überhaupt möglichen Ausprägungen reduziert wird. Bei der Erläuterung von Text-Mining Verfahren wurde bereits erwähnt, dass Listen von Akteure auf jene Einträge gekürzt werden können, deren Name tatsächlich im Text vorkommt. Ein Text, in welchem weder 'Merkel' noch 'Kanzlerin' oder 'Bundeskanzlerin' vorkommt, kann nicht über Angela Merkel sprechen. In diesem Fall könnte die Liste um diese Ausprägung gekürzt werden. Zu diesem Zweck muss jedoch technisch die Möglichkeit gegeben sein, dass eine Auswahlliste während der Codierung angepasst und dem Codierer nicht mit allen Ausprägungen präsentiert wird, die laut Codebuch möglich wären.

Situation 4: Der Codierer trifft eine unplausible Entscheidung

Bestimmt ein automatisches Verfahren die Wahrscheinlichkeit für eine bestimmte Ausprägung, so kann diese Information auch dann zur Unterstützung des Codierers herangezogen werden, wenn kein Vorschlag gemacht wird (z.B: weil es sich um eine lange Auswahlliste handelt oder keine eindeutig beste Lösung vorliegt). Hier kann die tatsächliche Entscheidung eines Codierers auf Plausibilität geprüft werden und der Codierer im Falle einer problematischen Entscheidung gewarnt werden. Wählt der Codierer also eine Ausprägung, für welche die Wahrscheinlichkeit sehr tief ist, so kann die Eingabemaske zur Vorbeugung von Flüchtigkeitsfehlern mit einer kurzen Rückfrage um Bestätigung dieser Entscheidung bitten. Für diese Art der Verknüpfung der beiden Komponenten der halbautomatischen Inhaltsanalyse muss die Möglichkeit geschaffen werden, eine Codierentscheidung vor der Speicherung auf ihre Plausibilität zu prüfen, um sie gegebenenfalls erst nach einer Bestätigung anzunehmen.

Bei dieser Art von Eingriffen in den Codierablauf ist zu bedenken, dass der Codierer durch solche Warnungen in seiner Arbeit behindert und möglicherweise verunsichert wird. Sie sollten daher nur dann zur Anwendung kommen, wenn die Wahrscheinlichkeit für eine tatsächliche Fehlentscheidung sehr hoch liegt.

Einschränkungen der halbautomatischen Inhaltsanalyse

Die oben skizzierte Erhebungsmethode der halbautomatischen Inhaltsanalyse verbindet die Effizienz automatischer Inhaltsanalyse mit dem Textverständnis menschlicher Codierer. Dabei werden dem Codierer einzelne Entscheidungen abgenommen und andere durch Vorschläge und eine Einschränkung der Freiheitsgrade erleichtert. Während die Methode auf diese Weise dafür sorgen kann, den kognitiven Aufwand des Codierers gering zu halten und seine Arbeitsabläufe zu beschleunigen, birgt sie auch Gefahren für die Qualität der Codierung.

Eine deutlich sichtbare Gefahr liegt in der möglichen Entscheidungsheuristik des Codierers, auf Vorschläge des Programms zu vertrauen. Obschon die Aufgabe des Codierers darin liegt, Texte systematisch und nach Vorgabe des Codebuches zu lesen und zu analysieren, zeigt sich in der Praxis, dass viele Codierer sich bei der Arbeit einfacher Entscheidungsheuristiken bedienen (Wirth, 2001). Vorschläge eines Programms zur automatischen Inhaltsanalyse sind dabei eine attraktive Heuristik, insbesondere wenn die Validität des Programms sich für einzelne Variablen als durchaus akzeptabel erweist. Gerade hier müssen Codierer gezielt geschult und zur Vorsicht gemahnt werden. Dies gilt sowohl für dichotome Kategorien, in welchen das Programm einen eindeutigen Vorschlag unterbreitet, als auch für die Hervorhebung besonders wahrscheinlicher Ausprägungen auf Auswahllisten. Auch in letzterem Fall kann der Codierer verleitet werden, nur die treffendste der vorgeschlagenen Kategorien zu wählen und nicht die gesamte Liste in Betracht zu ziehen.

Eine weitere Gefahr liegt in der Art der Fehler, welche ein Programm macht. Gerade statistische Verfahren wie die Bayes-Klassifikation neigen dazu, in unklaren Fällen den Modalwert als wahrscheinlichste Ausprägung zu ermitteln (Manning & Schütze, 1999). Gleichzeitig ist es bei diesen Verfahren möglich, dass beim Training des Verfahrens eine Stichprobe älterer Texte oder Texte aus einem anderen Medium vorliegen. Dies kann dazu führen, dass bestimmte Schlüsselworte in der Inhaltsanalyse zur Wahl einer falschen Ausprägung führen. Bei Text-Mining und Semantisch-Grammatikalischen Verfahren kann hingegen eine falsch definierte Regel dazu führen, dass ein bestimmter Fehler immer wieder gemacht und der Codierer immer wieder mit falschen Vorschlägen konfrontiert wird. Solche systematischen Fehler automatischer Verfahren können die Qualität der Daten maßgeblich beeinträchtigen. Hier müssen Codierer sensibilisiert werden und diese Fehler gewissenhaft korrigieren und gegebenenfalls die Projektleitung über Auffälligkeiten informieren.

Werden trainierte Verfahren angewandt, so ist zudem die Gefahr des fehlerhaften Trainings zu bedenken. Da das Programm aus allen Entscheidungen der menschlichen Codierer lernt, ist es besonders anfällig auf Fehler, welche diese machen. Zeichnet sich ein Codierer beim Training durch eine mangelhafte Validität aus, so wird das Programm mit Fehlentscheidungen trainiert und kann in der Folge den Codierer nicht durch Vorschläge unterstützen. Die Auswirkungen unsystematischer Fehlentscheidungen von Codierern sind hierbei weniger gravierend, da diese nur zu unschärferen Vorhersagen führen. Problematisch sind systematische Fehlentscheidungen, durch welche das Programm trainiert wird, eine falsche Vorhersage zu machen.

Zusammenfassend bedeutet dies, dass die halbautomatische Inhaltsanalyse nur dann funktionieren kann, wenn die computergestützten Verfahren im Hintergrund optimal vorbereitet werden und die Codierer gezielt geschult werden, die Vorschläge des Programms zu hinterfragen und zu kontrollieren. Zudem ist es ratsam, bei Reliabilitätstests der Codierer die Übereinstimmung mit

der Lösung der Codierleitung und mit der Lösung der automatischen Codierung zu prüfen. Liegt ein Codierer dort, wo sich die beiden Lösungen unterscheiden näher an der automatischen Lösung, so muss vermutet werden, dass er sich zu sehr auf die Vorhersage des Programms verlässt und diesbezüglich nachgeschult werden sollte.

Anwendungsbeispiel

Die in diesem Kapitel skizzierte halbautomatische Inhaltsanalyse wurde im Rahmen einer umfangreichen Inhaltsanalyse der Debatte von Arbeitslosigkeit in englischen Zeitungen in der Zeit der Stellenkürzungen im öffentlichen Raum im Herbst 2010 (siehe: NCCR 2013) getestet. Die Inhaltsanalyse untersuchte speziell das Framing der Arbeitslosigkeitsdebatte und wandte dabei eine analytische Erhebung von Frames an, bei welcher Problemdefinitionen, Ursachen, Folgen, Bewertungen und Massnahmen eines Problems getrennt erhoben werden (vgl. Matthes & Kohring, 2008). In jedem Artikel mussten hierfür mehrere Analyseeinheiten erkannt und codiert werden (Für eine ausführlichere Beschreibung, siehe: Wettstein, Wirth, Reichel, & Kühne, in diesem Band).

Für eine möglichst einfache Erfassung mehrerer Analyseeinheiten pro Artikel wurden die Daten über das Programm Angrist (Wettstein, 2013) direkt am Computer eingegeben. Das Programm bietet eine anpassungsfähige Eingabemaske, über welche mehrere Analyseebenen gleichzeitig bearbeitet werden können. Leichte Anpassungen der Eingabemaske auf die in Kapitel 0 umrissenen Anforderungen erlaubten es, das Programm für eine halbautomatische Inhaltsanalyse vorzubereiten. Zwei Codierer testeten die halbautomatische Inhaltsanalyse anschließend anhand von fünf Artikeln, welche bereits im Reliabilitätstest der Inhaltsanalyse zur Anwendung kamen und erstellten ein Gedankenprotokoll, um die Eignung der Methode abschätzen zu können.

Einbindung

Die halbautomatische Inhaltsanalyse wurde in diesem Anwendungsbeispiel mit einem einfachen Text-Mining für die automatische Erfassung der Formalia und einem naiven Bayes Klassifikator für die Generierung von Vorschlägen für Dichotome und multinominale Kategorien und die Einschränkung von Listen realisiert. Grammatisch-Semantische Verfahren kamen in diesem Beispiel nicht zum Einsatz.

Text Mining für Formalia

Da alle Texte für diese Inhaltsanalyse aus der Datenbank LexisNexis gewonnen wurden, wiesen sie ein beinahe[5] standardisiertes Format auf, bei welchem wichtige Informationen über den Artikel in

[5] Die Meta-Informationen der elektronischen Artikel aus der Datenbank LexisNexis befinden sich jeweils in den ersten und letzten Zeilen der Texte. Sie weisen ein Format auf, welches in den meisten Fällen für ein

Feldern vermerkt waren (Abbildung 1). Einfache Regeln erlaubten es, den Autor, die Rubrik, das Datum, die Seitenzahl, das Medium, die Länge und das Vorhandensein von Grafiken zu extrahieren und automatisch zu speichern. Der Codierer wurde bei diesen Entscheidungen nicht um eine Bestätigung gebeten.

Übermittlung des Textes an die automatische Inhaltsanalyse

In Angrist werden die bearbeiteten Texte direkt in die Eingabemaske geladen und neben den Formularen zur Eingabe von Daten eingeblendet. Für einen Teil der Verfahren wurde der gesamte Artikeltext an automatische Verfahren im Hintergrund übermittelt. Da jedoch mehrere Analyseeinheiten pro Artikel vorhanden sein können und das Programm jeweils nur die aktuell bearbeitete Einheit automatisch erfassen soll, wurde der Codierer gebeten, die aktuelle Analyseeinheit jeweils im Text zu markieren. Für jede Analyseeinheit wurde jeweils der ganze Abschnitt (resp. mehrere Abschnitte bei längeren Markierungen), in welchem die Markierung gemacht wurde, an die automatische Inhaltsanalyse übermittelt.

Text Mining für Akteurslisten

Für jede Analyseeinheit konnten mehrere politische Akteure codiert werden, welche zu diesem Thema Stellung nahmen. Um die Liste der Akteure übersichtlich zu halten, wurde dem Codierer eine reduzierte Liste angezeigt, welche nur jene Akteure enthielt, deren Namen oder Funktion im aktuell codierten Abschnitt vorkamen. Wurde ein Akteur nicht angezeigt, der codiert werden sollte (zum Beispiel aufgrund eines Spitznamens anstatt des echten Namens im Text), so konnte der Codierer sich die komplette Liste dennoch anzeigen lassen. In den meisten Fällen wurde die Liste auf 1-5 Akteure reduziert, wobei die korrekte Wahl in dieser kurzen Liste vertreten war.

Trainierte Klassifikation für Dichotome und multinominale Kategorien

Für Kategorien, bei welchen der Codierer zwischen zwei bis vier Ausprägungen wählen konnte, wurde ihm jeweils die plausibelste Entscheidung vorgeschlagen. Ermittelt wurde die Wahrscheinlichkeit für jede Ausprägung mittels Bayes'scher Klassifikation (Manning & Schütze, 1999), einem trainierten statistischen Verfahren, welches mit bisherigen Codierentscheidungen trainiert worden war. Da neben den Entscheidungen der Codierer auch der Text des jeweils bearbeiteten Abschnitts gespeichert wurde, konnte das Programm laufend mit neuen Codierungen

---

Medium in einem bestimmten Zeitraum einheitlich ist. Werden wie hier nur wenige Medientitel über einen Zeitraum von drei Monaten in die Analyse einbezogen, so ist die automatische Erfassung dieser Meta-Informationen mit wenigen Regeln möglich.

trainiert und in der Validität verbessert werden. Die Vorschläge waren in 87.22% der Fälle korrekt und ersparten damit dem Codierer den manuellen Eintrag seiner Codierung.

Für vier Kategorien, welche mehr als zehn Ausprägungen hatten, wurden die Codierentscheidungen über die Auswahl aus einer Liste erfasst. Auch hier wurde mittels Bayes'scher Klassifikation die Wahrscheinlichkeit für jede Ausprägung berechnet und die Liste wurde entsprechend des Ergebnisses angepasst. Es wurden jeweils die fünf wahrscheinlichsten Ausprägungen an den Anfang der Liste gesetzt, um dem Codierer eine lange Suche in der Liste zu ersparen. Um die Heuristik zu vermeiden, die erstplatzierte Ausprägung anzunehmen, wurden die Ergebnisse nicht nach Plausibilität sortiert sondern nach ihrer Reihenfolge im Codebuch. In 74.1% der Fälle war die richtige Wahl unter den fünf hervorgehobenen Ausprägungen.

Ergebnis

Fünf Artikel wurden von zwei Codierern mittels halbautomatischer Inhaltsanalyse erfasst, um die Methode in der Praxis zu testen. Die Erfahrungen wurden im Anschluss an jeden Artikel in einem kurzen Gedankenprotokoll festgehalten. Die zentralen Ergebnisse werden an dieser Stelle strukturiert zusammengefasst.

Bearbeitung des Textes am Bildschirm

Um der halbautomatischen Inhaltsanalyse die korrekten Textstellen zur aktuell bearbeiteten Analyseeinheit zu übermitteln, mussten die Codierer den Text direkt in der Eingabemaske lesen und die Analyseeinheiten markieren. Für beide Codierer war dieser Arbeitsablauf ungewohnt, da sie die Texte zuvor auf Papier gelesen und markiert hatten, um dann erst an der Eingabemaske ihre Daten einzugeben. Die Zeit, welche sowohl in der Vorbereitung der Inhaltsanalyse (kein Ausdrucken und Verteilen der Texte) und in der Erfassung (keine Übertragung der Markierungen in die Eingabemaske) eingespart werden kann, spricht dennoch für diese Art der Erfassung. Wenn in einem Projekt von Beginn an die Texte am Bildschirm bearbeitet werden, gewöhnen sich die Codierer schnell an diese Art, zu arbeiten[6].

Schneller Einstieg in den Artikel

Die Datenerfassung der manuellen Inhaltsanalyse begann jeweils mit sehr einfachen formalen Kategorien, anhand welcher sich der Codierer langsam auf den neuen Text einstellen kann. Indem zunächst die Länge des Textes, das Medium und Erscheinungsdatum eingetragen wurden, konnte

[6] Eine entsprechende Inhaltsanalyse wurde im Frühjahr 2013 durchgeführt. Das Angebot, die Texte in gedruckter Form zu erhalten, nahm keiner der 17 Codierer in Anspruch. Alle Codierer arbeiteten ausschliesslich am Bildschirm und verwendeten die Markierungswerkzeuge in Angrist.

man sich vom vorherigen Text lösen und sich auf den neuen einstellen. Da die Formalia in der halbautomatischen Inhaltsanalyse automatisch und ohne Bestätigung durch den Codierer erfasst wurden, begann die Arbeit am neuen Text mit dem Auffinden der Analyseeinheiten, also mit der ersten Aufgabe, welche nicht von automatischen Verfahren übernommen werden kann. Die Codierung 'einfacher' Kategorien zwischen zwei Artikeln, welche als angenehme Pause empfunden wurde, fiel damit weg. Gerade bei längeren Codiersitzungen könnte dies zu erhöhter Anstrengung führen und mehr Codierpausen nötig machen.

Vorschläge bei dichotomen und multinominalen Kategorien

Die Vorschläge wurden in den meisten Fällen als angenehme Hilfe empfunden, wenn sie sich mit den Entscheidungen des Codierers deckten. Bei der manuellen Inhaltsanalyse musste jede Entscheidung aus einem Dropdown-Menü oder mit Hilfe von Checkboxen angewählt werden, was jeweils mehrere Mausklicks pro Eingabeseite erforderte. Das Prüfen der Vorschläge war deutlich einfacher und war durch die hohe Trefferquote des trainierten Verfahrens sinnvoll.

In Fällen, in welchen sich die Vorschläge nicht mit den Entscheidungen des Codierers deckten, führten sie dazu, dass die betreffende Textstelle noch einmal gelesen wurde, um die eigene Entscheidung zu überdenken. Zwei Fälle wurden im Anschluss an die Codierung geschildert, in welchen dadurch eine Fehlentscheidung des Codierers vermieden werden konnte da der Vorschlag des Programms korrekt war. In diesen Fällen erhöhte also die halbautomatische Inhaltsanalyse die Validität der Codierung. In einigen Fällen war der Vorschlag des Programms jedoch falsch und führte nicht nur dazu, dass der Codierer die Textstelle erneut las und seine Entscheidung kritisch überdachte, sondern auch dazu, dass der Codierer darüber nachdachte, wie das Programm auf die Fehlentscheidung kam. Diese zusätzlichen Übelegungen führten zu einem höheren kognitiven Aufwand und einer leicht erhöhten Bearbeitungszeit dieser Kategorie.

Einschränkung der Listen

Die Einschränkung langer Listen von Ausprägungen durch trainierte Verfahren und Textmining wurde mehrheitlich als Hilfe empfunden. Da die hervorgehobenen Ausprägungen oder die angebotenen Akteure meist die korrekte Auswahl enthielten, konnten diese Kategorien deutlich schneller bearbeitet werden. In den wenigen Fällen, in welchen die Auswahl die korrekte Ausprägung nicht enthielt, wurde die eigene Entscheidung jedoch angezweifelt und anhand des Textes noch einmal überprüft.

Bei Akteurslisten wurde keine Bayes'sche Klassifiketion verwendet, sondern die Liste wurde durch Text-Mining auf die möglicherweise genannten Akteure beschränkt. Hier sorgten insbesondere unbekannte Akteure (also jene, welche auch auf der kompletten Liste fehlen) für Probleme. In diesen Fällen wussten die Codierer nicht, ob der Akteur über die Restausprägung ('sonstiger Akteur') erfasst

werden soll und suchten den Akteur auf der Gesamtliste. Diese Fälle waren aufgrund der kurzen Liste von relevanten Akteuren in dieser Inhaltsanalyse sehr häufig, wodurch dieser Teil der halbautomatischen Inhaltsanalyse nicht geschätzt wurde.

Im Allgemeinen wurde die halbautomatische Erfassung der Texte und die Arbeit an der Eingabemaske aber positiv beschrieben und die Codierer fühlten sich gerade durch die Vorschläge bei multinominalen Kategorien sehr gut unterstützt. Für diese Kategorien sank auch die Bearbeitungszeit erheblich, was die Erfassung der Texte insgesamt um fast 20% beschleunigte. Dabei ist jedoch zu bedenken, dass die Codierer dieselben Texte bereits ein Jahr zuvor im Rahmen des Reliabilitätstests für diese Inhaltsanalyse bearbeitet hatten und sich möglicherweise an einzelne Entscheidungen erinnern konnten.

Fazit

Die hier vorgestellte halbautomatische Inhaltsanalyse hat zwei grundsätzliche Ziele. Einerseits soll die Effizienz menschlicher Codierer dadurch erhöht werden, dass eine im Hintergrund arbeitende automatische Inhaltsanalyse Entscheidungen erleichtert oder abnimmt, andererseits sollen systematische Fehler der automatischen Inhaltsanalyse dadurch vermieden werden, dass ein menschlicher Codierer mit Sprachkompetenz und Kontextwissen Entscheidungen prüft. Der erste Praxistest dieser Methode weist auf eine Erfüllung beider Ziele hin. Die Codierer nahmen die Vorschläge des Programms sehr schnell an, wenn diese ihren eigenen Entscheidungen entsprachen und hinterfragten sowohl ihre eigenen als auch die automatischen Entscheidungen, falls sie nicht übereinstimmten. Dies erhöhte nicht nur die Validität der automatischen Codierung (durch manuelle Korrekturen derselben), sondern auch die Validität der manuellen Codierung, welche in kritischen Fällen ebenfalls hinterfragt und korrigiert wurde.

Die Geschwindigkeit, mit welcher die Codierer die Entscheidungen geprüft haben, lässt jedoch auch aufhorchen, da damit möglicherweise Fehler des automatischen Verfahrens angenommen wurden, welche der Codierer ohne Unterstützung nicht gemacht hätte. Um dies zu verhindern ist es wichtig, dass die bereits angemerkten Fallstricke der halbautomatischen Inhaltsanalyse umgangen und die Codierer optimal auf ihre Arbeit mit dieser Methode geschult werden. Gerade die Möglichkeit, Vorschläge in ambigen Situationen zu unterdrücken und damit den Codierer zu einer Enscheidung zu zwingen, ist mit Nachdruck empfohlen. Zudem empfiehlt es sich auch, nur jene Kategorien durch ein trainiertes Verfahren vorherzubestimmen, welche mit mehr als 90% Präzision vorherbestimmt werden können. Im vorliegenden Test galt dies nur für 18 der 32 Kategorien. Für Kategorien, bei welchen die Irrtumswahrscheinlichkeit größer ist, sollten keine Vorschläge gemacht

und die Entscheidung komplett dem Codierer überlassen werden, da falsche Vorhersagen den Codierprozess eher verlangsamen und kognitiven Aufwand bedeuten.

Ein weiteres interessantes Ergebnis des Praxistests ist die Unsicherheit, welche durch unvollständige Akteurslisten entsteht, die durch das Programm weiter gekürzt werden. Hier sieht sich der Codierer mit falschen Auswahllisten konfrontiert, welche eher verwirren als die Auswahl erleichtern. Der Grund für dieses Problem liegt darin, dass für die manuelle Inhaltsanalyse auf eine möglichst kurze und übersichtliche Akteursliste Wert gelegt wurde, welche durch den Codierer schnell erfasst werden kann. Für die halbautomatische Inhaltsanalyse wäre es jedoch von Vorteil eine möglichst umfassende Akteursliste zu definieren und diese jeweils durch Textmining einzuschränken, so dass der Codierer aus einer kurzen Liste wählen kann, welche den korrekten Eintrag mit hoher Wahrscheinlichkeit enthält. Eine ebenfalls sinnvolle Möglichkeit sind hier auch Listen, welche durch die Eingabe eines Namens durchsucht werden können.

Ausblick

Die in diesem Kapitel geschilderte Methode hat sich in einem ersten Praxistest als sinnvolle Ergänzung manueller Inhaltsanalysen erwiesen, da sie nicht nur die Effizienz, sondern auch die Qualität der Codierung erhöhen kann. Die ersten Ergebnisse lassen jedoch auch einige Fragen offen. So ist unklar, wie gründlich die Codierer die Vorschläge des Programms prüfen, bevor sie sie akzeptieren und welchen Einfluss ein falscher Vorschlag auf die Verarbeitung des Textes haben kann. Ebenso konnte in diesem Praxistest die tatsächliche Zeitersparnis der halbautomatischen Inhaltsanalyse nicht getestet werden, da die Codierer die Texte bereits kannten. Dies bedeutet auch, dass der Einfluss der halbautomatischen Erhebung auf die Validität der Codierung nicht beziffert werden kann. Hier konnten lediglich anekdotische Fälle als Hinweis gewertet werden.

Klarheit zu den einzelnen Punkten könnte ein kontrolliertes Experiment schaffen, bei welchem mehrere Codierer mittels manueller und halbautomatischer Inhaltsanalyse eine Reihe unbekannter Texte codieren. Denkbar ist in diesem Zusammenhang auch eine gründliche Beobachtung des Codierverhaltens mittels Eye-Tracking, um die Reflektion der auotmatisch generierten Vorschläge zu untersuchen.

Kurzfristig erlaubt der Input, welchen der Praxistest dieser Methode geliefert hat, die halbautomatische Inhaltsanalyse weiter zu entwickeln und die Einbindung automatischer Verfahren in Angrist zu verbessern. Dazu gehört auch die Entwicklung neuer Abfrage-Elemente, welche speziell auf halbautomatische Erhebungen abgestimmt sind. Das Ziel ist es, Angrist optimal für eine halbautomatische Inhaltsanalyse einzurichten, um künftig eine frei verfügbare Software-Lösung für diese Erhebungsmethode anbieten zu können.

Literatur

Colleoni, E., Rozza, A., & Arvidsson, A. (2014). Echo Chamber or Public Sphere?: Predicting Political Orientation and Measuring Political Homophily in Twitter Using Big Data. Journal of Communication, n/a. doi:10.1111/jcom.12084

Conway, M. (2006). The Subjective Precision of Computers: A Methodological Comparison with Human Coding in Content Analysis. *Journalism & Mass Communication Quarterly*, *83*(1), 186–200. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=ufh&AN=21184758&site=ehost-live

Crawley, C. E. (2007). Localized Debates of Agricultural Biotechnology in Community Newspapers: A Quantitative Content Analysis of Media Frames and Sources. *Science Communication*, *28*(3), 314–346.

Danowski, J. (1993). Network Analysis of Message Content. In W. D. Richards & G. A. Barnett (Eds.), *Progress in communication sciences: Vol. 12. Progress in Communication Sciences* (pp. 197–220). Norwood, NJ: Ablex.

Früh, W. (2009). *Inhaltsanalyse: Theorie und Praxis* (1st ed., Vol. 2501). Konstanz: UVK Verl.-Ges.

Hart, R. P. *Diction 5.0*. Retrieved from http://rhetorica.net/diction.htm

Kim, L. (2011). Media framing of stem cell research: A cross-national analysis of political representation of science between the UK and South Korea. *Journal of Science Communication*, *10*(3), 1–16.

Landmann, J., & Züll, C. (2004). Computerunterstützte Inhaltsanalyse ohne Diktionär?: Ein Praxistest. *ZUMA-Nachrichten*, *54*(28), 117–140.

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, Mass: MIT Press.

Matthes, J., & Kohring, M. (2008). The Content Analysis of Media Frames: Toward Improving Reliability and Validity. *Journal of Communication*, *58*(2), 258–279. doi:10.1111/j.1460-2466.2008.00384.x

Miller, M., & Riechert, B. P. (2001). Frame Mapping: A Quantitative Method for Investigating Issues in the Public Sphere. In M. D. West (Ed.), *Progress in communication sciences: Vol. 16. Theory, method, and practice in computer content analysis* (pp. 61–75). Westport, Conn: Ablex Publ.

NCCR. (2013). *Module 4: Changing processes and strategies of political participation and representation: comparing public debates*. Retrieved from http://www.nccr-democracy.uzh.ch/research/phaseII/module-4

Roberts, C. W. (1989). Other than Counting Words: A Linguistics Approach to Content Analysis. *Social Forces*, *68*(1), 147–177.

Scharkow, M. (2012). *Automatische Inhaltsanalyse und maschinelles Lernen*. Dissertation Universität Hohenheim (1st ed.). Berlin: epubli GmbH.

Schneider, G., Clematide, S., & Rinaldi, F. (2011). Detection of interaction articles and experimental methods in biomedical literature. *BMC Bioinformatics*, *12*(Suppl 8), S13. doi:10.1186/1471-2105-12-S8-S13

Sjøvaag, H., Moe, H., & Stavelin, E. (2012). Public Service News on the Web: A Large-Scale Content Analysis of the Norwegian Broadcasting Corporation's Online News. Journalism Studies, 13(1), 90–106. doi:10.1080/1461670X.2011.578940

Stewart, C. O., Pitts, M. J., & Osborne, H. (2011). Mediated Intergroup Conflict: The Discursive Construction of "Illegal Immigrants" in a Regional U.S. Newspaper. *Journal of Language and Social Psychology*, *30*(1), 8–27. doi:10.1177/0261927X10387099

Stuckardt, R. (2000). *Qualitative Inhaltsanalyse durch Computer: Ein uneinlösbarer Anspruch?* Dissertation an der Johann Wolfgang Goethe Universität zu Frankfurt am Main. *Tenea Wissenschaft*. Berlin, Frankfurt (Main): Tenea.

van Atteveldt, W., Kleinnijenhuis, J., & Ruigrok, N. (2008). Parsing, Semantic Networks, and Political Authority: Using Syntactic Analysis to Extract Semantic Relations from Dutch Newspaper Articles. *Political Analysis*, *16*(4), 428–446. doi:10.1093/pan/mpn006

Wettstein, M. (2012). Term-Mapping zur komparativen Analyse öffentlicher Debatten: Eine Anwendung der Smallest Space Analysis für Inhaltsanalysen. In B. Stark, M. Magin, O. Jandura, & M. Maurer (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 8. Methodische Herausforderungen Komparativer Forschungsansätze* (pp. 138–157). Köln: Herbert von Halem Verlag.

Wettstein, M. (2013). *angrist 1.0: Dokumentation und Anleitung für die Programmierung des Codierer-Interface*. Retrieved from http://www.ipmz.uzh.ch/Abteilungen/Medienpsychologie/Recource/Angrist.html

Wettstein, M., Wirth, W., Reichel, K., & Kühne, R. (In Press). Zum Problem der Mehrfachcodierung: Sind drei wirklich genug?: Eine systematische Fehleranalyse. In K. Sommer, W.

Wirth, M. Wettstein, & J. Matthes (Eds.), *[Methoden und Forschungslogik der Kommunikationswissenschaft, 9].* Köln: Herbert von Halem Verlag.

Wirth, W. (2001). Der Codierprozess als gelenkte Rezeption: Bausteine für eine Theorie des Codierens. In W. Wirth & E. Lauf (Eds.), *Inhaltsanalyse. Perspektiven, Probleme, Potentiale* (pp. 157–182). Köln: Halem.

Wüest, B., Clematide, S., Bünzli, A., Laupper, D., & Frey, T. (2011). Electoral Campaigns and Relation Mining: Extracting Semantic Network Data from Newspaper Articles. *Journal of Information Technology & Politics*, *8*(4), 444–463. doi:10.1080/19331681.2011.567387

Young, L., & Soroka, S. (2012). Affective News: The Automated Coding of Sentiment in Political Texts. *Political Communication*, *29*(2), 205–231. doi:10.1080/10584609.2012.671234

Tages-Anzeiger	**Le Monde**
Dienstag 19. Oktober 2010	**16 novembre 2010 mardi**
Die Grünliberalen beweisen, dass auch die Mitte Mode sein kann	**Croissance, déficits, chômage : la France n'est pas encore vraiment sortie de la crise**
AUTOR: Friedli Daniel; Von Daniel Friedli	**AUTEUR: Philippe Le Coeur**
RUBRIK: SCHWEIZ; NaN; S. 5	**RUBRIQUE: CONTRE-ENQUETE; Pg. 16**
LÄNGE: 893 Wörter	**LONGUEUR: 819 mots**
[Text]	**[Text]**

Abbildung 1: Standardisierte Darstellung von Informationen zum Artikel in Texten aus LexisNexis. Die Felder sind sprachabhängig unterschiedlich benannt und unterscheiden sich leicht in der Formatierung der Inhalte. Für jedes Medium können jedoch klare Regeln für Text-Mining Algorithmen aufgestellt werden.

Anhang A3: Quantitative Ursachenbestimmung medialer Aufmerksamkeitsschübe

veröffentlicht als:

Wettstein, M. (2015). Quantitative Ursachenbestimmung medialer Aufmerksamkeitsschübe. Publizistik, 60(3), 325–343. doi:10.1007/s11616-015-0238-4

Quantitative Ursachenbestimmung medialer Aufmerksamkeitsschübe

[Bitte nicht aus diesem Manuskript zitieren. Manuskript zur Publikation: Wettstein, M. (2015). Quantitative Ursachenbestimmung medialer Aufmerksamkeitsschübe. Publizistik, 60(3), 325–343.]

Zusammenfassung

Die Medienberichterstattung über langfristig aktuelle Themen wie Arbeitslosigkeit, Umweltschutz oder Migration ist geprägt von unregelmäßigen Aufmerksamkeitsschüben, welche ein Thema jeweils für kurze Zeit prominent machen. Über diese Phasen erhöhter Aufmerksamkeit kann sowohl die öffentlich wahrgenommene Wichtigkeit einer Debatte, als auch der Druck auf politische Entscheidungsträger erhöht werden. Die Frage, wer für diese Aufmerksamkeitsschübe verantwortlich ist und damit die Macht hat, die öffentliche Agenda zu bestimmen, ist deswegen von großer gesellschaftlicher Relevanz und kann im Einzelfall über eine qualitative Analyse des Medieninhalts und der politischen Debatte bestimmt werden.

In diesem Beitrag wird nach einer Möglichkeit gesucht, die Ursachen medialer Aufmerksamkeitsschübe quantitativ zu unterscheiden, um qualitative Analysen untermauern und Debatten auch über einen längeren Zeitraum nachvollziehbar beschreiben zu können. Mit der zeitlichen Abfolge der Fokussierung auf bestimmte Themen und Akteure vor und während eines Aufmerksamkeitsschubes wird ein Merkmal gefunden, welches von der Ursache der erhöhten Medienaufmerksamkeit abhängen kann. Zusammenhänge zwischen dem Verlauf eines Aufmerksamkeitsschubes und dessen Ursachen werden anhand der Daten einer Inhaltsanalyse der Arbeitslosigkeitsdebatte in drei Ländern nachgewiesen und können der Entwicklung quantitativer Analysen von Debatten als Grundlage dienen.

Schlüsselworte: Öffentliche Debatte; Medienaufmerksamkeit; Fokussierung; Quantitative Inhaltsanalyse

Abstract

Media coverage of public debates has been shown to exhibit long phases of reduced media attention which are eventually interrupted by sudden attention bursts. These attention bursts increase the salience and perceived importance of an issue in the public and may exert an influence on political decisions. Therefore, the question for the causes of these attention bursts and for their initiators is vitally important to understand the role and power of mass media in political communication. In single cases, the causes of increased media coverage may be investigated by means of qualitative content analysis to assess their exact causes and circumstances.

In this paper, a quantitative approach to the investigation of attention bursts is demonstrated which may help to identify the initiators and circumstances of an attention burst without profound knowledge of a debate. It is assumed that the focus on issues and actors during attention bursts is influenced by the cause of media attention. To test these assumptions, a time series analysis of attention bursts in a content analysis on the debate on unemployment in three countries is performed. The results indicate that the temporal change of focus and volume of news coverage betrays the causes of media attention bursts and may be used to devise methods for the longitudinal and comparative analysis of media attention.

Keywords: Public Debate; Attention Cycle; Media Focus; Quantitative Content Analysis

Forschungsinteresse

Massenmedien widmen ihre Aufmerksamkeit einer Vielzahl unterschiedlicher Themen mit gesellschaftlicher Relevanz. Das Maß an Aufmerksamkeit, welche einem einzelnen Thema wie dem Umweltschutz oder der Arbeitslosigkeit zukommt, ist jedoch nicht konstant über die Zeit. Im Gegenteil ist zu beobachten, dass der Fokus der Massenmedien regelmäßig ändert, und dass sich jeweils unterschiedliche Themen an die Spitze der Medienagenda setzen. Für den Verlauf einzelner Debatten resultiert daraus ein typisches Muster aus Routinephasen mit konstant tiefer Medienaufmerksamkeit (vgl. Kepplinger, 2001, S. 123), welche durch kurze Schübe intensiver Berichterstattung unterbrochen werden (vgl. Barabási 2005). In den kurzen Zeiträumen erhöhter Medienberichterstattung werden aktuelle und vergangene Ereignisse diskutiert und eingeordnet, und die Standpunkte und Forderungen politischer und zivilgesellschaftlicher Akteure publiziert und diskutiert (vgl. Brossard et al. 2004; Kepplinger und Habermeier 1995; Zhu 1992).

Die Berichterstattung während medialer Aufmerksamkeitsschübe zeichnet sich durch zwei besondere Eigenschaften aus: Zum einen ist die Berichterstattung in dieser kurzen Zeit deutlich intensiver als das die Anzahl tatsächlicher Ereignisse nahelegen würde (vgl. Vasterman 2005; Kepplinger 2011, S. 93-94). Zum anderen durchdringt die Berichterstattung das Mediensystem und ist damit für Leser unterschiedlicher Medienangebote erfahrbar (vgl. Djerf-Pierre 2012; Waldherr 2014).

Bisherige Untersuchungen zu den Ursachen medialer Aufmerksamkeitsschübe konnten zeigen, dass sich die Auslöser für erhöhte Aufmerksamkeit von Fall zu Fall unterscheiden. So wurden unvermittelte Schlüsselereignisse wie Katastrophen oder unerwartete Ankündigungen (vgl. Brosius und Eps 1995), absichtlich herbeigeführte und inszenierte Ereignisse zur Generierung von Aufmerksamkeit (vgl. Imhof und Eisenegger 1999; Pfetsch 2003), politische Debatten und Entscheidungen (vgl. Jarren und Donges 2002) oder gezielte Kampagnen einzelner Medienangebote und Verlage (vgl. Schiffer 2006) als mögliche Ursachen identifiziert. Eine Unterscheidung zwischen diesen Ursachen bei einzelnen Aufmerksamkeitsschüben kann dabei von grosser Bedeutung sein.

Einerseits ergibt sich eine hohe gesellschaftliche Relevanz der Ursachenbestimmung medialer Aufmerksamkeitsschübe aus dem Agenda-Setting Potential der Massenmedien. Die erhöhte Medienaufmerksamkeit für ein Thema beeinflusst nicht nur die wahrgenommene Wichtigkeit dieses Themas unter den Rezipienten, sondern kann auch Druck auf das politische System ausüben (vgl. Rogers und Dearing 1988; Tan und Weaver 2007). Die Frage nach Akteuren, welche die Häufigkeit und Intensität von Aufmerksamkeitsschüben beeinflussen können, ist damit für die Analyse der Dynamik öffentlicher Meinung von zentraler Bedeutung. Andererseits wird insbesondere bei Themen mit globaler Reichweite das internationale Agenda-Setting zunehmend zu einem Gegenstand der

Forschung (vgl. Schäfer et al. 2011). In diesem Zusammenhang ist es nicht nur wichtig, gleichzeitige intensive Berichterstattung über ein Thema in mehreren Ländern zu identifizieren, sondern auch ihre Entstehung auf Ereignisse mit globaler Reichweite oder internationales Agenda-Setting zurückführen zu können.

Die Ursache eines Aufmerksamkeitsschubs muss jedoch jeweils im Einzelfall bestimmt werden und ist meist nur nach einer vertieften qualitativen Analyse der Medienberichterstattung oder durch Befragungen von Journalisten und Experten möglich. Während sich dieser Zugang für Einzelfälle durchaus anbietet, ist er für extensive Analysen im Längsschnitt oder im Ländervergleich meist zu aufwändig und stellt jeweils eine subjektive Interpretation des Forschers dar, deren Qualität und Nachvollziehbarkeit vom Vorgehen und der Gründlichkeit bei der Durchführung abhängt (vgl. Schwab-Trapp 2003, S. 178-179). Um eine qualitative Interpretation der Hintergründe einer Debatte empirisch zu untermauern oder um den Aufwand weitreichender Analysen zu verringern, sind quantitativ messbare Indikatoren notwendig, anhand derer die Ursache von Aufmerksamkeitsschüben nachvollziehbar bestimmt werden können. Gerade für umfangreiche Untersuchungen sind dabei Indikatoren wünschenswert, welche sich auch computergestützt erheben lassen und damit den Aufwand einer solchen Analyse erheblich senken.

Um solche Indikatoren zu finden soll in diesem Beitrag der Frage nachgegangen werden, durch welche Merkmale sich Aufmerksamkeitsschübe mit unterschiedlichen Ursachen oder Urhebern quantitativ unterscheiden. Hierfür soll zunächst auf den allgemeinen Verlauf und bekannte Ursachen medialer Aufmerksamkeitsschübe eingegangen werden, um messbare Eigenschaften der Berichterstattung herzuleiten, die für die Unterscheidung von verschiedenen Typen von Aufmerksamkeitsschüben relevant sind. Annahmen, welche sich aus dieser theoretischen Aufarbeitung ergeben, werden im Anschluss über eine Sekundäranalyse von drei Inhaltsanalysen öffentlicher Debatten überprüft.

Theoretische Fundierung

In der Kommunikationswissenschaft haben sich mit der Erforschung der Auswirkungen von Schlüsselereignissen auf die Berichterstattung und der Interdependenz zwischen dem politischen System und den Massenmedien zwei Forschungstraditionen herausgebildet, welche das Phänomen der gesteigerten Medienaufmerksamkeit aus unterschiedlichen Perspektiven beleuchten. In diesem Beitrag soll eine Synthese aus diesen beiden Perspektiven hergestellt werden, um daraus konkrete Hypothesen zu quantitativen Unterscheidungsmerkmalen zwischen Aufmerksamkeitsschüben mit unterschiedlichen Auslösern abzuleiten.

Aufmerksamkeitsschub durch journalistische Routinen

Aus einer handlungstheoretischen Perspektive wird die Allokation medialer Aufmerksamkeit als Reaktion von Journalisten, Redaktionen und Rezipienten auf konkrete Ereignisse untersucht. In diesem Zusammenhang wurde gezeigt, dass gesellschaftlich relevante Schlüsselereignisse selbstverstärkende Rückkopplungsprozesse auslösen können, welche innert weniger Tage zu einer gesteigerten Aufmerksamkeit für einen bestimmten Themenbereich führen (vgl. Kepplinger 2011, S. 87). Das Resultat ist ein sprunghafter Anstieg des Volumens der Berichterstattung auf ein Niveau, welches im Vergleich zur Anzahl relevanter Ereignisse überproportional ist (vgl. Vasterman 2005, S. 509).

Die Interaktionen zwischen Journalisten und Rezipienten, welche für die Rückkopplungsprozesse und die stark gesteigerte Aufmerksamkeit verantwortlich sind, wurden detailliert beschrieben (vgl. Kepplinger, 2001; Vasterman 2005) und konnten bereits in Simulationsstudien nachvollzogen werden (vgl. Miltner & Waldherr, 2013). Der Prozess beginnt jeweils mit einem Ereignis, welches eine anfängliche Berichterstattung anstößt und das Interesse sowohl der Journalisten als auch der Rezipienten für dieses Thema stimuliert (vgl. Vasterman 2005, S. 513; Kepplinger, 2001: 123). Durch diese anfängliche öffentliche Aufmerksamkeit steigt der Nachrichtenwert für Ereignisse und Themen, welche mit dem anfänglichen Schlüsselereignis in Zusammenhang stehen (vgl. Kepplinger und Habermeier 1995, S. 381; Keller 2003, S. 401). In der Folge intensiviert sich die Berichterstattung über Themen im Zusammenhang mit dem Schlüsselereignis und das Interesse der Rezipienten wird erneut gesteigert. Die Folgeberichterstattung löst sich schließlich vom eigentlichen Schlüsselereignis und widmet sich neuen, themenverwandten Ereignissen. Gleichzeitig setzt vor allem in Qualitätsmedien schnell eine "Tiefenberichterstattung" (Kepplinger 2001, S. 127) ein, in deren Verlauf Bezüge zu anderen Themenbereichen oder Ereignissen in der Vergangenheit hergestellt und Meinungen von Experten publiziert werden.

Durch die intensive Berichterstattung werden schließlich Reaktionen von Interessengruppen stimuliert, deren Forderungen einen Bezug zum aktuellen Thema haben (vgl. Kepplinger 2011, S. 87). Die Berichterstattung wird damit nicht nur thematisch breiter, sondern zeichnet sich auch durch eine zunehmende Akteursvielfalt aus. Erst wenn die Informationen zum Thema erschöpft sind oder ein themenfremder medialer Aufmerksamkeitsschub die öffentliche Aufmerksamkeit bindet, endet der Prozess, und die Berichterstattung geht wieder in eine weniger intensive Routinephase über (vgl. Zhu 1992).

Kepplinger (2001: 125) unterscheidet zwischen drei unterschiedlichen Typen von Ereignissen, welche als Schlüsselereignisse fungieren und diese Rückkopplungsprozesse auslösen können. *Genuine* und *mediatisierte* Ereignisse sind Ereignisse, welche auch ohne das Vorhandensein von

Massenmedien eingetreten wären. Während genuine Ereignisse wie Katastrophen oder Unfälle komplett unabhängig von der Medienberichterstattung ablaufen und unerwartet eintreten, sind mediatisierte Ereignisse wie Sportanlässe oder angekündigte Veröffentlichungen von Ergebnissen absehbar und auf eine anschließende Berichterstattung ausgerichtet (vgl. Kepplinger, 2001: 125). Der dritte Typ von Ereignissen, die *inszenierten* Ereignisse oder Pseudo-Ereignisse (vgl. Boorstin 1961), werden von Akteuren bewusst initiiert um Berichterstattung zu provozieren. Diese Ereignisse umfassen Störaktionen, Demonstrationen oder Anschläge, welche die öffentliche Aufmerksamkeit auf ein Thema oder einen Akteur lenken (vgl. Kepplinger, 2001: 125). Inszenierte Ereignisse sind insbesondere bei nicht-etablierten Akteuren ein beliebtes Mittel, um auf die eigene Agenda aufmerksam zu machen (vgl. Imhof & Eisenegger 1999, S. 208-211) und werden entsprechend von Pressemitteilungen und aktiver Medienarbeit seitens der Initianten begleitet.

In einzelnen Fällen kann es auch gelingen, den Rückkopplungsprozess hinter einem Aufmerksamkeitsschub ohne ein konkretes Schlüsselereignis auszulösen. Dies geschieht dann, wenn durch die Kumulation von Berichten zu einer Entwicklung oder einem Missstand durch einzelne Journalisten die öffentliche Aufmerksamkeit so stark angeregt wird, dass auch konkurrierende Medienangebote das Thema aufgreifen. Dieses Cross-Media Agenda-Setting (vgl. Schiffer 2006) führt zu weniger abrupten Aufmerksamkeitsschüben, da die Medienlandschaft nur langsam durchdrungen wird, kann aber vereinzelt nach intensiven Kampagnen in der alternativen Presse (vgl. Mathes und Pfetsch 1991) oder einzelnen Online-Medien und Blogs (vgl. Schiffer 2006) beobachtet werden.

Aufmerksamkeitsschub als Teil der publizistischen Funktion

Aus einer systemtheoretischen Perspektive lässt sich aus der Verschränkung des politischen Systems und des Mediensystems im Verlauf politischer Debatten ein anderer Typ von Aufmerksamkeitsschüben ableiten. Hier reagieren nicht individuelle Journalisten auf einzelne Ereignisse, sondern das Mediensystem reagiert - um seine Funktion als intermediärer Akteur (vgl. Jarren und Donges 2002, S. 138-139) zu erfüllen - auf politische Debatten. Diese Debatten intensivieren sich von der Feststellung eines Problems über die Behandlung im politischen System bis zu einer Entscheidung (vgl. Downs 1972), was auch zu einer stetigen Intensivierung der Berichterstattung führt. Dies ist insbesondere bei emotionalen oder konfliktreichen Debatten zu erwarten, welche das öffentliche Interesse anzuregen vermögen (vgl. Nisbet und Huge 2006).

Mediale Aufmerksamkeitsschübe als Spiegel politischer Debatten stellen eine funktionelle und gewünschte Anpassung des Mediensystems an politische Entscheidungsfindungsprozesse dar (vgl. Downs 1972; Fowler et al. 2012). Sie sind außerdem vorhersehbar und planbar, wenn die politische Tagesordnung bekannt ist. Dadurch zeichnen sie sich – im Gegensatz zur oben beschrieben Reaktion auf Schlüsselereignisse – durch einen langsamen und thematisch fokussierten Anstieg der

Berichterstattung aus. Erst durch eine konkrete politische Entscheidung, deren Verkündung als Schlüsselereignis fungieren kann, wird die Berichterstattung schließlich sprunghaft intensiver und thematisch vielfältiger.

Forschungsfrage und Hypothesen

Aus den beiden Perspektiven, welche die Reaktion medialer Berichterstattung auf Ereignisse, Kampagnen und politische Prozesse beschreiben, lassen sich zum einen konkrete Ursachen und Urheber für einzelne Aufmerksamkeitsschübe ableiten, zum anderen können auch Annahmen über den Verlauf der Berichterstattung getroffen werden. Der Verlauf der Medienaufmerksamkeit, also das Gesamtvolumen der Berichterstattung, unterscheidet sich in erster Linie in Hinsicht auf die Intensität und die Geschwindigkeit der Veränderung, was eine quantitative Bestimmung erschweren kann. Geeignet für eine quantitative Unterscheidung ist hingegen die Fokussierung auf Unterthemen und Akteure im Zeitverlauf, weil diese vorhersehbar auf unterschiedliche Auslöser für erhöhte Medienaufmerksamkeit reagiert. Dies betrifft sowohl die Richtung als auch den Zeitpunkt der Fokussierung der Berichterstattung vor und während eines Aufmerksamkeitsschubes.

So weist Vasterman (2005) darauf hin, dass in der anfänglichen Orientierungsphase nach einem Schlüsselereignis ein breites Spektrum an Experten zum Thema befragt wird. Im Verlauf des Aufmerksamkeitsschubs öffnet sich die Arena weiter, wenn auch Interessenverbände und politische Akteure aktiv an die Öffentlichkeit wenden (vgl. Kepplinger 2001, S. 87). Dies gilt jedoch vor allem für genuine Ereignisse, welche sowohl für Journalisten als auch für zentrale Akteure überraschend eintreten. Bei mediatisierten Schlüsselereignissen können zentrale Akteure bereits vor dem Ereignis durch gezielte Pressearbeit Aufmerksamkeit auf sich ziehen und sich für vorgängige Stellungnahmen zur Verfügung stellen. Auch bei inszenierten Ereignissen bereiten sich die Initianten auf die Medienberichterstattung und ihre Pressearbeit vor. Gerade bei Inszenierungen, welche überraschend oder schockierend wirken sollen, wird die Pressearbeit aber nicht vor dem Ereignis aufgenommen. Die Fokussierung auf Akteure wird deswegen bei genuinen Ereignissen schnell sinken, während sie zu Beginn und im Vorfeld mediatisierter und inszenierter Ereignisse stärker sein kann und erst allmählich sinkt, sobald weitere Akteure sich am Diskurs beteiligen. Bei Kampagnen, welche ohne Schlüsselereignis zu einem Aufmerksamkeitsschub führen, lässt sich kein typischer Verlauf der Akteursfokussierung skizzieren, da diese Kampagnen unterschiedlich stark gegen einzelne Akteure und ihr Verhalten gerichtet sein können.

Während politischer Debatten kann angenommen werden, dass die Fokussierung auf Akteure bereits im Vorfeld des Aufmerksamkeitsschubs sinkt, da in der Berichterstattung über politische Debatten meist mehrere politische Lager präsent sind. Im Verlauf der Debatte kann die

Akteursvielfalt durch den Einbezug von Experten und die Vermittlung zivilgesellschaftlicher Interessen noch weiter zunehmen.

Zusammenfassend lassen sich vier Hypothesen zum Verlauf der Fokussierung auf einzelne Akteure formulieren (siehe Abbildung 1):

Hypothese 1: Aufmerksamkeitsschübe als Reaktion auf genuine Ereignisse zeichnen sich durch eine sinkende Akteursfokussierung aus.

Hypothese 2: Aufmerksamkeitsschübe als Reaktion auf mediatisierte Ereignisse zeichnen sich durch eine vorgängige Akteursfokussierung aus.

Hypothese 3: Aufmerksamkeitsschübe als Reaktion auf inszenierte Ereignisse zeichnen sich durch eine ansteigende Akteursfokussierung aus.

Hypothese 4: Aufmerksamkeitsschübe als Reaktion auf politische Debatten zeichnen sich durch eine vorgängige Vielfalt von Akteuren aus.

Auch für die thematische Fokussierung während eines Aufmerksamkeitsschubs können aus der Beschreibung der festgestellten Dynamiken konkrete Annahmen abgeleitet werden. Da Schlüsselereignisse bereits kurz nach ihrem Eintreten aus unterschiedlichen Perspektiven beleuchtet und mit verschiedenen Themen in Verbindung gebracht werden (vgl. Kepplinger 2001, S. 127), kann hier von einer schnellen thematischen Defokussierung ausgegangen werden. Eine solche Defokussierung kann auch dann erwartet werden, wenn ein Aufmerksamkeitsschub nicht durch ein Schlüsselereignis, sondern durch eine journalistische Kampagne ausgelöst wird. In diesem Fall ist jedoch zusätzlich von einer steigenden thematischen Fokussierung im Vorfeld des Aufmerksamkeitsschubes auszugehen, während das Thema von verschiedenen Medienangeboten aufgegriffen wird.

Für die Berichterstattung über politische Debatten kann angenommen werden, dass sie bereits im Vorfeld des Aufmerksamkeitsschubes auf das spezifische politische Problemfeld fokussiert ist, da bereits in der Frühphase einzelne Standpunkte zu der bevorstehenden Debatte publiziert werden. Erst nach Erreichen einer Entscheidung wird die Berichterstattung thematisch vielfältiger, wenn die Reichweite und Auswirkungen der Entscheidung öffentlich diskutiert und analysiert werden.

Für die thematische Fokussierung von Aufmerksamkeitsschüben lassen sich entsprechend zwei Hypothesen formulieren (siehe Abbildung 1):

Hypothese 5: Aufmerksamkeitsschübe als Reaktion auf genuine, mediatisierte oder inszenierte Ereignisse zeichnen sich durch eine hohe thematische Vielfalt aus

Hypothese 6: Aufmerksamkeitsschübe als Reaktion auf Kampagnen oder politische Debatten zeichnen sich durch eine vorgängige thematische Fokussierung aus.

In Abbildung 1 sind fünf unterschiedliche Szenarien skizziert, welche die aufgestellten Hypothesen zusammenfassend darstellen. Die Darstellung zeigt deutlich, wie aus der Kombination der Verläufe der Akteurs- und Themenfokussierung klar unterscheidbare idealtypische Verläufe folgen. Bestätigen sich diese Verläufe in realen Aufmerksamkeitsschüben, so ermöglicht diese Aufstellung die Bestimmung der Ursache medialer Aufmerksamkeit über die quantitative Erhebung der Fokussierung und des Volumens der Berichterstattung.

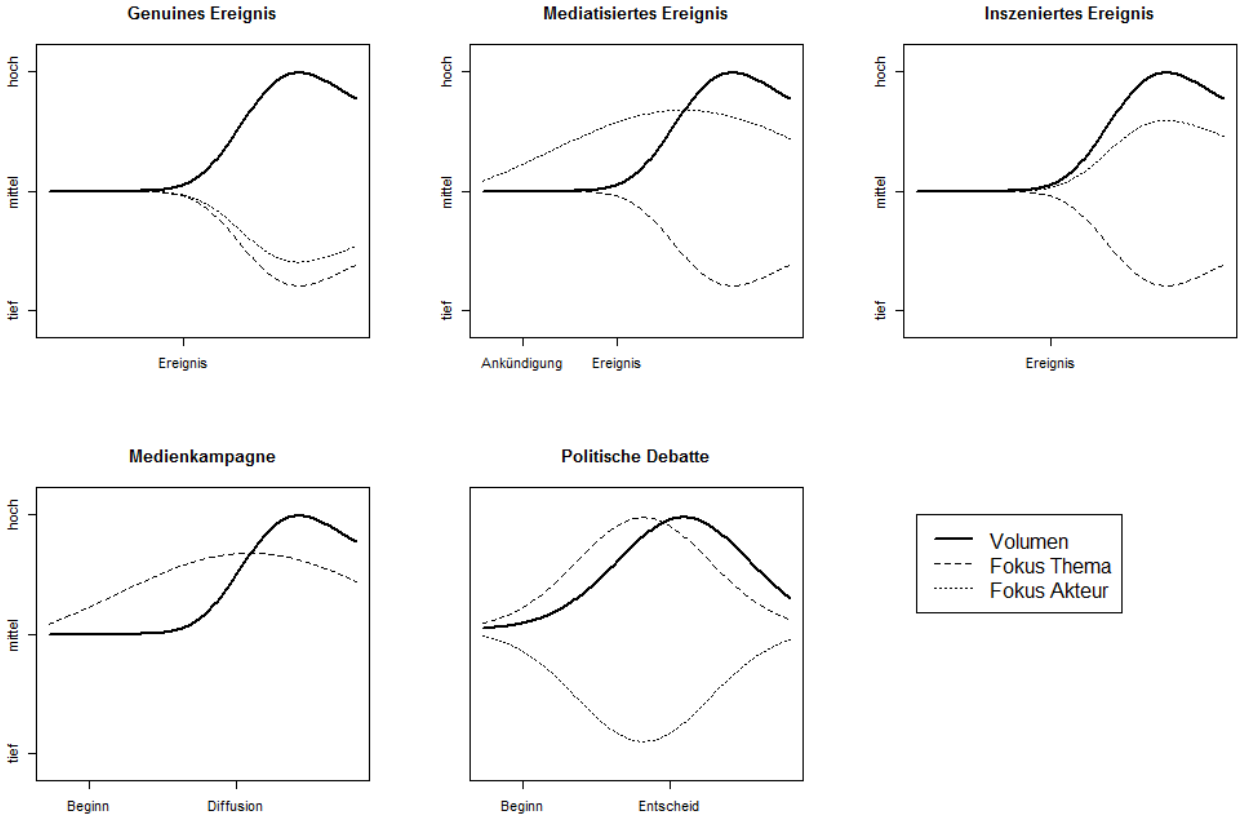


Abbildung 22: Hypothetischer Zusammenhang zwischen der Ursache eines Aufmerksamkeitsschubs und dem zeitlichen Verlauf des Berichterstattungsvolumens und der Fokussierung auf Themen und Akteure. Für alle Grafiken wird vor dem Aufmerksamkeitsschub ein mittleres Volumen und eine durchschnittlich starke Fokussierung angenommen. In realen Situationen können jedoch Niveauunterschiede auftreten.

Methode

Um die Hypothesen zum Zusammenhang zwischen der Ursache und dem Verlauf von Aufmerksamkeitsschüben zu prüfen, wird in dieser Studie eine Sekundäranalyse einer Inhaltsanalyse durchgeführt, welche die Inhalte von fünf Aufmerksamkeitsschüben in drei Ländern erfasst hat. Für diese Fallbeispiele liegen neben der quantitativen Analyse der Inhalte auch qualitative Analysen zu

den einzelnen Aufmerksamkeitsschüben vor. Dies macht sie für eine erste Überprüfung der angenommenen Zusammenhänge zu geeigneten Beispielen.

Bei den untersuchten Medieninhalten handelt es sich um die Berichterstattung über die Arbeitslosigkeitsdebatte in Deutschland, Großbritannien und Dänemark zwischen dem 10. September und 10. Dezember 2010. In diesem Quartal erholte sich die Wirtschaft langsam von der Bankenkrise, welche im Sommer 2007 begonnen hatte (vgl. International Monetary Fund 2009). Die Situation auf dem Arbeitsmarkt und in den Staatshaushalten war jedoch weiterhin angespannt. Dies senkte die Schwelle für Medienaufmerksamkeit für das Thema Arbeitslosigkeit, wodurch mit dem Auftreten von Aufmerksamkeitsschüben jederzeit gerechnet werden konnte. Die Auswahl für diese drei europäischen Länder wurde getroffen, da in allen drei Ländern fast zeitgleich ein Ereignis im Zusammenhang mit der Arbeitslosigkeitsdebatte auftrat. In Deutschland wurden am 30. Oktober die tiefsten Arbeitslosenzahlen seit 18 Jahren vermeldet, in England wurde am 20. Oktober die Kürzung von 500'000 Stellen im öffentlichen Dienst angekündigt, und in Dänemark wurde am 27. Oktober die Streichung von 3000 Stellen beim Windradhersteller *Vestas* bekannt. Die Ereignisse haben damit eine unterschiedliche Vorgeschichte, Intensität und Valenz.

Eine Längsschnittanalyse des Berichterstattungsvolumens bestätigt für alle drei Ereignisse einen Aufmerksamkeitsschub und weist zudem auf zwei weitere Schübe im untersuchten Zeitraum hin, welche in diesem Beitrag ebenfalls als Fallbeispiele untersucht werden. Für die Prüfung der Hypothesen wird zum einen eine qualitative Analyse der Berichterstattung zu jedem Aufmerksamkeitsschub durchgeführt, zum anderen werden die quantitativen Daten im Hinblick auf Berichterstattungsvolumen, Akteursfokussierung und thematische Fokussierung ausgewertet.

Quantitative Analyse

In allen drei Ländern wurde im Zeitraum zwischen dem 10. September und 10. Dezember 2010 eine quantitative Inhaltsanalyse durchgeführt. In jedem Land wurden Artikel zum Thema Arbeitslosigkeit in neun bis elf Zeitungen (siehe Tabelle 1) untersucht. Um relevante Artikel zum Thema zu finden, wurden in jeder Sprache sechs bis zehn Schlüsselbegriffe definiert, nach welchen die Zeitungen in Online-Datenbanken oder in den E-Papers durchsucht wurden. Anschließend wurden manuell jene Artikel aussortiert, in welchen die Begriffe nur in einem themenfremden Kontext wie Sportberichterstattung, Gerichtsfällen oder Werbung auftraten. Aus den verbleibenden Artikeln wurde je nach Umfang der Berichterstattung in den einzelnen Titeln eine Zufallsstichprobe von 40-100% manuell analysiert.

Tabelle 5: Übersicht über Zeitungen und Zeitschriften in der Inhaltsanalyse.

Zeitung	Artikel Gesamt	Codierte Artikel	Stichprobe	Suchbegriffe
Berliner Zeitung	310	133	43%	*arbeitslos*, *beschäftigung*, job*,

BILD / BAMS	197	78	40%	arbeit, kurzarbeit*, hartz, grundsicherung, regelsätze, minijob, stellen
Berliner Morgenpost	468	196	42%	
B.Z.	332	145	44%	
FAZ (+Sonntag)	601	238	40%	
Spiegel	67	33	49%	
Süddeutsche Zeitung	711	711	100%	
Die Welt (+Sonntag)	368	154	42%	
Die Zeit	110	48	44%	
The Daily Mail (+Sunday)	665	265	40%	unemploy*, job*, "jobseekers allowance", "spending review", housing benefit, redundancy, dismiss*
The Daily Telegraph (+Sunday)	600	295	49%	
London Evening Standard	350	147	42%	
The Guardian	1198	705	59%	
The Independent	287	287	100%	
The Daily Mirror	1006	401	40%	
The Observer	100	46	46%	
The Sun (+News of the World)	104	48	46%	
24 Timer	35	25	71%	arbejdsløs*, beskaeftigel*, jobmulighed*, langtidsledig, ledig, praktikplads
Politiken	412	281	68%	
Urban	74	49	66%	
B.T.	89	61	69%	
Berlingske Nyhedsmagasin	23	23	100%	
Berlingske Tidende	501	348	69%	
Ekstra Bladet	174	118	68%	
Jyllands-Posten	491	337	69%	
Metro Xpress	108	70	65%	
Norrebro Avis	10	8	80%	
Osterbro Avis	4	4	100%	

Anmerkung: Ausgewiesen ist jeweils die Gesamtzahl gefundener Artikel, sowie die Anzahl und der prozentuale Anteil codierter Artikel.
In jedem Land wurden für alle Zeitungen dieselben Suchbegriffe verwendet, um Artikel zu finden. Die Gesamtanzahl bezieht sich auf die Artikel, in welchen die Suchbegriffe im Zusammenhang mit Arbeitslosigkeit genannt wurden. Alle themenfremden Artikel wurden manuell aussortiert.

In der manuellen Analyse der Zeitungstexte wurden alle Aussagen im Zusammenhang mit Arbeitslosigkeit erfasst. Insgesamt 14 Codierer bearbeiteten die Zeitungsartikel, deren Sprache sie entweder als Muttersprache oder fließend gesprochene Fremdsprache beherrschten. Sie bestimmten für jede Aussage zum Thema Arbeitslosigkeit ihren Urheber, sowie ihren Inhalt. Für die inhaltliche Codierung wurde ein Codebuch nach dem Vorbild von Matthes und Kohring (2008) verwendet, welches ein Kategoriensystem mit 118 unterschiedlichen Problemdefinitionen, Ursachen und Folgen von Arbeitslosigkeit, sowie möglichen Maßnahmen gegen Arbeitslosigkeit vorgab. Die Codierer mussten jede Aussage in diese Kategorien einordnen.

Die Qualität der Erhebung wurde in einem verdeckten Reliabilitätstest untersucht, in welchem jedem Codierer während der Dauer der Erhebung 88 Test-Artikel zugewiesen wurden. Die Texte wurden zufällig unter die gewöhnlichen Zeitungsartikel gemischt und enthielten keinen Hinweis auf eine Test-Situation. In diesem Test erreichten die Codierer eine durchschnittliche paarweise Übereinstimmung von 87.3%. Der zufallsbereinigte Reliabilitätskoeffizient Scott's Pi (vgl. Scott 1955) betrug 0.691, was auf eine sehr gute Reliabilität der Codierungen hinweist (vgl. Lombard et al. 2002).

Das Volumen der Berichterstattung wurde über die Anzahl codierter Aussagen zum Thema Arbeitslosigkeit pro Tag über alle Medien operationalisiert. Dadurch wird nicht nur eine Zunahme von Artikeln, sondern auch eine Zunahme ihrer Länge und ihrer Ausführlichkeit als Indikator für eine erhöhte Medienaufmerksamkeit gewertet. Um für die reduzierte Berichterstattung am Wochenende, sowie zufällige Varianz zwischen Wochentagen zu korrigieren, wurde das Volumen für jeden Tag als Durchschnitt eines gleitenden Fensters von 7 Tagen berechnet.

Für die Berechnung der Fokussierung von Medienberichterstattung schlägt Boydstun (2008, S.181) das Maß der negativen standardisierten Entropie vor. Für jeden Tag der Berichterstattung wird der Anteil p_i der N potentiell vorkommenden Elemente ermittelt. Mit Hilfe dieser Anteile lässt sich berechnen, wie stark die Berichterstattung sich auf einzelne Elemente fokussiert (siehe Formel 1).

$$Fokus = 1 - \frac{-\sum_i^N p_i * \ln(p_i)}{\ln(N)} \qquad (1)$$

Die Fokussierung, welche auf diese Weise berechnet wird, kann Werte zwischen 0 und 1 annehmen, wobei 0 für eine perfekte Gleichverteilung und 1 für eine Fokussierung auf genau ein Element steht. Für die Fokussierung auf Akteure wurden die Anteile der unterschiedlichen Sprecher verwendet, welche sich an jedem Tag zum Thema Arbeitslosigkeit äußerten. Die thematische Fokussierung wurde über die Verteilung der codierten Problemdefinitionen, Folgen, Ursachen und Maßnahmen gemessen. Die potentiell mögliche Anzahl an Akteuren oder Framekomponenten war dabei für jedes Land die Gesamtanzahl unterschiedlicher codierter Elemente. Auch für die Berechnung der Fokussierung wurde ein gleitendes Fenster von 7 Tagen verwendet, um die Varianz innerhalb einer Woche zu korrigieren und die Verläufe mit dem Verlauf des Berichterstattungsvolumens vergleichbar zu machen.

Sowohl für Spitzenwerte des Volumens als auch der Fokussierung der Berichterstattung wurden lokale Minima und Maxima in den errechneten Zeitreihen gesucht. Werte wurden dann als lokales Maximum oder Minimum betrachtet, wenn sie in einem 7-Tages Durchschnitt höher als die beiden benachbarten Werte und gleichzeitig in einem äußeren Quartil (d.h. höher oder tiefer als 75% der gemessenen Werte) lagen. Auf diese Weise wurden insgesamt fünf lokale Maxima des Berichterstattungsvolumens identifiziert, welche die in dieser Studie untersuchten Aufmerksamkeitsschübe darstellen.

Qualitative Inhaltsanalyse

Für die qualitative Inhaltsanalyse wurde jeweils der Zeitraum drei Tage vor und nach einem Berichterstattungsmaximum betrachtet. In jedem Untersuchungszeitraum wurde in einem Leitmedium nach Artikeln gesucht, welche über die Arbeitslosigkeitsdebatte, über das Verhalten einzelner Akteure in dieser Debatte, oder über konkrete Ereignisse berichteten. Für die Analyse

wurden in einem ersten Schritt alle Artikel zur Arbeitslosigkeitsdebatte in den betrachteten sieben Tagen anhand derselben Suchbegriffe wie in der quantitativen Inhaltsanalyse gesucht. Für Deutschland wurde die *Süddeutsche Zeitung*, für England *The Guardian* und für Dänemark die Zeitung *Politiken* analysiert.

In einem zweiten Schritt wurden die Artikel gelesen und nach Äußerungen über die Medienberichterstattung, einzelne Akteure und konkrete Ereignisse durchsucht, um sie in Oberkategorien der Berichterstattung und Metaberichterstattung einzuteilen. Die Texte, welche sich zu diesen Themen äußerten, wurden nach der von Mayring (2007, S. 59-74) beschriebenen qualitativen Zusammenfassung von Texten paraphrasiert und auf ihre Aussagen zur Arbeitslosigkeitsdebatte reduziert. Diese Zusammenfassung, welche im Ergebnisteil verkürzt dargestellt ist, wurde schließlich verwendet, um die Ursachen der einzelnen Aufmerksamkeitsschübe zu eruieren.

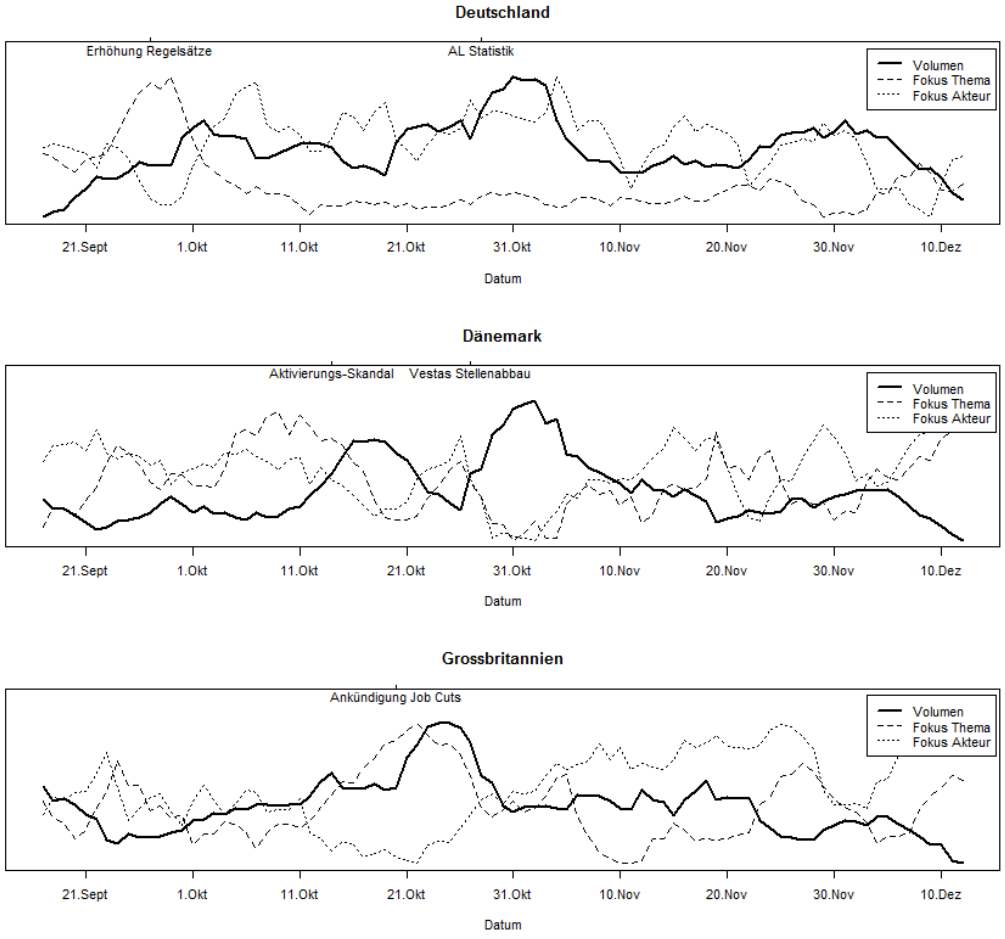


Abbildung 23: Darstellung des Verlaufs der Arbeitslosigkeitsdebatte in Deutschland, Dänemark und Großbritannien.

Für jede Debatte sind der Verlauf des Berichterstattungsvolumens, sowie der Verlauf der Fokussierung auf einzelne Unterthemen der Debatte und einzelne Akteure dargestellt. Die Verläufe sind an ihrer empirischen Spannweite standardisiert.

Ergebnisse

Die Quantitative Analyse zeigte im Untersuchungszeitraum fünf Aufmerksamkeitsschübe an, in welchen die Anzahl der Aussagen zum Thema Arbeitslosigkeit einen kurzfristigen Höhepunkt erreichten (siehe Tabelle 2). Für jeden dieser Aufmerksamkeitsschübe wurde eine qualitative Inhaltsanalyse durchgeführt, um die jeweilige Ursache zu bestimmen. Ausgehend von dieser Analyse wurden die Hypothesen zur Fokussierung auf Themen und Akteure getestet, welche sich auf die Ursachen beziehen. Die Hypothesen werden dabei nicht entlang ihrer oben eingeführten Nummerierung abgearbeitet, sondern nach der Reihenfolge der untersuchten Aufmerksamkeitsschübe und deren Ursachen.

Tabelle 6: Zeitpunkte der Aufmerksamkeitsschübe und Zeiträume für die qualitative Inhaltsanalyse ihrer Ursachen und Begleitumstände.

	Maximale Berichterstattung	Zeitraum qualitative Inhaltsanalyse	Anzahl Artikel
Deutschland 1	3.10.2010	30.9. – 6.10.2010	62
Deutschland 2	31.10.2010	28.10. – 3.11.2010	39
Dänemark 1	18.10.2010	15.10. – 21.10.2010	38
Dänemark 2	1.11.2010	28.10. – 4.11.2010	46
Grossbritannien 1	24.10.2010	21.10. – 27.10.2010	81

Deutschland

Die Berichterstattung während des ersten Aufmerksamkeitsschubs in Deutschland, welcher seinen Höhepunkt am 3. Oktober 2010 erreichte, war stark auf die Entscheidung der Bundesregierung fokussiert, die Regelsätze für die Arbeitslosenversicherung Hartz IV um fünf Euro anzuheben und zweckgebundene Bildungsbeiträge einzuführen. Mit dieser Entscheidung am 27. Oktober 2010 ging die Bundesregierung auf ein Urteil des Bundesverfassungsgerichts vom Februar 2010 (BVG 2010) ein, welches zu einer politischen Debatte über die Höhe und Berechnung der Arbeitslosenentschädigung führte. Die Berichterstattung widmete sich den Gründen für die Entscheidung, der Kritik an der Höhe der Anpassung durch Opposition und Gewerkschaften, sowie den Folgen der Entscheidung für einzelne Arbeitslose und das Budget. Das Thema der Regelsatzerhöhungen und ihrer Auswirkungen war über den gesamten Erhebungszeitraum der qualitativen Untersuchung prominent.

Neben diesem prominenten Thema wurde besonders zu Beginn des Aufmerksamkeitsschubes am 30.September und 1.Oktober verstärkt über die Arbeitslosenstatistik des Monats September berichtet. Diese wurde für einzelne Regionen in Bayern separat ausgewiesen und diskutiert. Die Artikel über die Entwicklung der Arbeitslosenquote wiesen jedoch jeweils nur wenige Aussagen auf, wodurch sie weniger stark zur Berichterstattung beitrugen als die Regelsatzdebatte.

Am Rande wurden weitere Themen im Zusammenhang mit Arbeitslosigkeit berichtet, darunter Einzelschicksale betroffener Personen, Armut und Überschuldung als Folgen von Arbeitslosigkeit, die Auswirkungen der niedrigen Arbeitslosenquote auf den Staatshaushalt und die Arbeitslosigkeit im Ausland. Das zentrale Thema - und damit der Auslöser des Aufmerksamkeitsschubes in dieser Zeitspanne – war jedoch die Regelsatzdebatte im Bundestag und die Entscheidung der Bundesregierung. Damit handelt es sich in diesem Fall um einen Aufmerksamkeitsschub als Folge einer politischen Debatte.

Hypothesen 4 und 6 sagten für die Berichterstattung im Vorfeld dieser Art von Aufmerksamkeitsschüben eine thematische Fokussierung und eine Defokussierung bezüglich der Akteure voraus. Ein Höhepunkt der thematischen Fokussierung am 27. September und ein Tiefpunkt der Akteursfokussierung am 28. September bestätigen beide Hypothesen.

Die Berichterstattung zum zweiten Aufmerksamkeitsschub, welcher am 31. Oktober 2010 seinen Höhepunkt erreichte, enthielt erneut positive Meldungen zum Rückgang der Arbeitslosigkeit in einzelnen Regionen. Diese wurden jedoch von der intensiven Berichterstattung über die nationale Arbeitslosenstatistik, die Umstände ihrer Veröffentlichung und ihre Berechnungsgrundlage überschattet. Am 29. Oktober berichtete die Arbeitsministerin Ursula von der Leyen, dass die Arbeitslosenquote die psychisch bedeutende Grenze von drei Millionen unterschritten und den tiefsten Stand seit 18 Jahren erreicht hat. Die Art der Veröffentlichung der Arbeitslosenzahlen wurde in der Berichterstattung kritisiert und mit der Reaktion des Wirtschaftsministers Rainer Brüderle auf andere positive Meldungen zur Konjunkturlage in diesem Herbst in Verbindung gebracht:

> *"Die Bekanntgabe wäre der Bundesagentur für Arbeit überlassen worden, wie immer zum Monatsende. Doch nun ist die Zahl auf einen Wert gesunken, der haarscharf unter drei Millionen liegt. Eine Zwei vor dem Komma! Die niedrigste Zahl seit 18 Jahren! Und das auch noch zum Jahrestag der Koalition. Klar, dass Ursula von der Leyen eine solche Nachricht selbst verbreiten will; klar, dass Rainer Brüderle, der Kollege aus dem Wirtschaftsministerium, ihr die Show nicht alleine überlässt."* (Esslinger 2010, S. 4).

Neben der Kritik an der Art der Veröffentlichung wurden auch die Folgen der niedrigen Arbeitslosenquote für Leih- und Kurzarbeiter, den Fachkräftemangel und eine mögliche Vollbeschäftigung in einzelnen Regionen diskutiert. Andere Themen im untersuchten Zeitraum umfassten die hohe Arbeitslosenquote in den USA, Einzelschicksale arbeitsloser Personen, die Förderung und Ausbildung jugendlicher Erwerbsloser und in einzelnen Fällen erneut die Erhöhung der Hartz-IV Regelsätze. Der Auslöser für die kurzfristig erhöhte Aufmerksamkeit war jedoch die

medienwirksam zelebrierte Veröffentlichung der Arbeitslosenstatistik, welche als mediatisiertes Ereignis betrachtet werden kann.

Hypothese 2 sagte bei dieser Art von Aufmerksamkeitsschub eine vorangehende Akteursfokussierung voraus. Diese Annahme wird durch lokale Maxima am 17. und 30. Oktober bestätigt. Hypothese 5, welche für Aufmerksamkeitsschübe nach Schlüsselereignissen generell eine größere thematische Vielfalt vorhersagt, wird jedoch durch die Daten nicht bestätigt. Zwar ist die thematische Fokussierung während dieses Aufmerksamkeitsschubes sehr tief, ein lokales Minimum ist jedoch erst nach Abklingen der Aufmerksamkeit am 6. November auszumachen.

Dänemark

In Dänemark hatte die Zeitung *Berlingske Tidende* im September und Oktober 2010 die Vermittlung und Aktivierung Arbeitsloser durch kommunale Behörden in einer mehrwöchigen Kampagne unter dem Schlagwort *Jobcircus* kritisiert. Im Zuge dieser Kampagne wurde gegen die Kommune Kopenhagen der Vorwurf der Misswirtschaft mit Steuergeldern laut, welcher vom Ombudsmann untersucht wurde. Auf die journalistische Kampagne und das Schlussgutachten des Ombudsmanns, welches am 14. Oktober veröffentlicht wurde und Kopenhagen entlastete, nahm die Zeitung *Politiken* besonders zu Beginn des Aufmerksamkeitsschubs Bezug. Dabei wurde aber weniger auf eine mögliche Verschwendung von Steuergeldern eingegangen, sondern auf die Aktivierung unvermittelbarer (dän: *ikkearbejdsmarkedsparate*) Arbeitsloser fokussiert, welche im Gutachten als unnötig und möglicherweise schädlich für Betroffene bezeichnet wurde. Die Unterscheidung zwischen vermittelbaren und unvermittelbaren Arbeitslosen, die positiven Seiten der Aktivierungspolitik für einzelne Branchen, sowie die Diskussion von Einzelschicksalen waren in der ersten Hälfte des Aufmerksamkeitsschubes prominente Themen. Danach verlagerte sich die Berichterstattung auf Auslandsberichterstattung und einzelne Berichte über arbeitslose Personen in anderen Zusammenhängen.

Der Grund für den Aufmerksamkeitsschub, welcher am 14. Oktober durch einen Anstieg der Berichterstattung beginnt und am 18. Oktober seinen Höhepunkt erreicht, liegt in der Veröffentlichung des Gutachtens, welches zu den Anschuldigungen der Misswirtschaft Stellung bezog und den Höhepunkt einer Medienkampagne einer einzelnen Zeitung darstellte. Hypothese 6 sagt für diese Art von Aufmerksamkeitsschüben eine vorangehende thematische Fokussierung voraus und wird durch einen Höhepunkt der thematischen Fokussierung am 9. Oktober bestätigt.

Ein zweiter Aufmerksamkeitsschub der Arbeitslosigkeitsdebatte erreichte seinen Höhepunkt am 1. November, nachdem der Windrad-Produzent *Vestas* am 27. Oktober die Streichung von 3000 Stellen in Dänemark bekanntgegeben hatte. Mehrere Artikel widmeten sich während dieses

Aufmerksamkeitsschubes den wirtschaftlichen Folgen für die Firma und das Land, der wirtschaftlichen Situation der betroffenen Gebiete, den betroffenen Personen und der intensiven Berichterstattung selber. Neben diesem zentralen Thema wurde über die Vorteile der Aktivierung Arbeitsloser, die globale wirtschaftliche Situation und die Arbeitslosigkeit im Ausland, insbesondere den USA, berichtet.

Der Auslöser für die hohe Medienaufmerksamkeit war in diesem Fall eine überraschende Ankündigung von Stellenkürzungen, welche als genuines Schlüsselereignis eingeordnet werden kann. Für diesen Typ von Aufmerksamkeitsschüben sagen Hypothesen 1 und 5 eine gleichzeitige Defokussierung bezüglich Akteuren und Themen voraus. Die Hypothesen werden durch Minima der Akteurs- und Themenfokussierung am 1. November 2010 bestätigt.

Großbritannien

Der Höhepunkt der Berichterstattung zur Arbeitslosendebatte Großbritanniens am 24.Oktober 2010 folgte vier Tage nach der Bekanntgabe der Kürzung von 500'000 Stellen im öffentlichen Dienst durch den britischen Kanzler George Osborne. Diese Kürzungen – welche den im Vorfeld vermuteten Umfang deutlich überstiegen – und ihre Folgen waren das zentrale Thema der Berichterstattung des *Guardian*. In diesem Zusammenhang wurde über mögliche Streiks in Teilen des öffentlichen Dienstes, die Auswirkungen der Kürzungen auf Schulen, öffentliche Bibliotheken und den öffentlichen Nahverkehr, sowie Geschichten über einzelne Betroffene berichtet. Gleichzeitig wurden die Entscheidung der Regierung und das Verhalten ihrer Mitglieder von unterschiedlichen Seiten kritisiert.

Die erhöhte Aufmerksamkeit für die Arbeitslosigkeitsdebatte wurde demnach durch eine politische Entscheidung ausgelöst und sollte gemäß Hypothesen 4 und 6 bereits im Vorfeld eine geringe Akteursfokussierung und eine starke thematische Fokussierung aufweisen. Beide Hypothesen werden durch ein lokales Minimum der Akteursfokussierung am 19. Oktober und ein lokales Maximum der thematischen Fokussierung am 22. Oktober bestätigt.

Diskussion

In diesem Beitrag wurde nach quantitativen Merkmalen medialer Aufmerksamkeitsschübe gesucht, welche es ermöglichen, ihre Ursachen und Entstehung ohne tiefgreifendes Vorwissen zur untersuchten Debatte induktiv zu unterscheiden. Während sich Aufmerksamkeitsschübe bezüglich des Verlaufs des Berichterstattungsvolumens nur schwach unterscheiden, wurden mit der Fokussierung auf Unterthemen und Akteure zwei Eigenschaften gefunden, welche von den Ursachen von Aufmerksamkeitsschüben abhängen können und Rückschlüsse auf die Entstehungsumstände zulassen.

Für fünf zentrale Ursachen kurzfristig erhöhter Medienaufmerksamkeit wurden Annahmen für den Verlauf der Fokussierung der Berichterstattung hergeleitet und empirisch geprüft. Die Befunde zu fünf beispielhaften Aufmerksamkeitsschüben in der Arbeitslosigkeitsdebatte in Europa im Herbst 2010 bestätigen die getroffenen Annahmen und illustrieren die unterschiedlichen Verläufe von Aufmerksamkeitsschüben als Reaktion auf verschiedene Ursachen. Eine quantitative Auswertung der Fokussierung auf Themen und Akteure im Zeitverlauf kann damit als Grundlage für die induktive Unterscheidung von Aufmerksamkeitsschüben herangezogen werden.

Die idealtypischen Verläufe, welche als Reaktion auf unterschiedliche Ursachen erhöhter Medienaufmerksamkeit skizziert wurden, trafen nur in einem Fall nicht zu, in welchem die thematische Fokussierung nach einem mediatisierten Schlüsselereignis nicht wie erwartet sank, sondern auf einem tiefen Niveau konstant blieb. Dennoch zeigt sich in den Verläufen der Fokussierung und des Volumens der Berichterstattung eine Möglichkeit der quantitativen Unterscheidung der Ursachen medialer Aufmerksamkeitsschübe. Dabei muss jedoch eingeräumt werden, dass der empirische Nachweis der angenommenen Zusammenhänge über nur fünf Beispiele noch sehr schwach ist und eine Prüfung der steigenden Akteursfokussierung nach inszenierten Ereignissen (Hypothese 3) schuldig bleibt.

Zudem erwies sich die Bestimmung der Muster über die Abfolge lokaler Minima und Maxima zwar als ausreichend für eine erste Überprüfung der angenommenen Zusammenhänge, stellt sich jedoch unter Umständen als ein zu wenig differenzierendes Instrument heraus. So ist es mit dieser Methode nicht möglich, die Geschwindigkeit der Veränderung von Kennwerten zu bestimmen, welche ebenfalls als Unterscheidungsmerkmal zwischen Mustern dienen kann. Zeitreihenanalysen könnten eine differenziertere Beschreibung von Aufmerksamkeitsschüben erlauben, auch wenn diese keine idealtypischen Verläufe zeigen oder zu schwach sind, um als eindeutiges lokales Maximum identifiziert zu werden.

Bei der quantitativen Beschreibung von Debatten aufgrund von Volumen und Fokussierung der Berichterstattung sollte zudem bedacht werden, dass die hier vorgestellten Muster nur idealisierte Verläufe isolierter Aufmerksamkeitsschübe darstellen. Überlagern sich unterschiedliche Ursachen für Medienaufmerksamkeit wie zum Beispiel eine politische Kampagne und ein unvorhergesehenes Ereignis, so werden die beobachteten Verläufe ungleich komplexer und können zu falschen Schlüssen verleiten. Eine solche Vermischung konnte im Fall des ersten Aufmerksamkeitsschubes in Dänemark beobachtet werden, in welchem eine Medienkampagne erst durch eine Pressekonferenz breite Beachtung fand. Die quantitative Analyse sollte daher zumindest von einer oberflächlichen inhaltlichen Auseinandersetzung mit der Debatte begleitet werden.

Fazit und Ausblick

In der Medienberichterstattung öffentlicher Debatten sind regelmäßige, kurze Schübe erhöhter Aufmerksamkeit zu beobachten, welche die wahrgenommene Relevanz eines Themas, sowohl in der Bevölkerung als auch in der Politik, erhöhen können. Als Ursachen für diese Aufmerksamkeitsschübe kommen unvermittelte Ereignisse, aber auch politische Debatten, Aktionen und Pressearbeit politischer Akteure oder journalistische Kampagnen in einzelnen Medienangeboten in Frage. Um die Funktion der Massenmedien in einer Debatte verstehen, und den Einfluss politischer Kräfte und zufälliger Ereignisse abschätzen zu können, ist die Identifikation der Ursache einzelner Aufmerksamkeitsschübe von großer Bedeutung.

In diesem Beitrag wurde eine Möglichkeit aufgezeigt, genau diese Unterscheidung anhand quantitativer Merkmale der Berichterstattung vorzunehmen. Es konnte gezeigt werden, dass der Verlauf der Fokussierung auf Akteure und Themen vor und während eines Aufmerksamkeitsschubes von den Ursachen der erhöhten Medienaufmerksamkeit abhängt. Dieser Befund erlaubt es, Methoden zur quantitativen Beschreibung von Aufmerksamkeitsschüben in öffentlichen Debatten zu entwickeln.

Ein methodischer Zugang, welcher in dieser Hinsicht besonders interessant werden kann, ist die automatische Analyse von Aufmerksamkeitsschüben. Bereits heute sind Anwendungen zur automatischen Erkennung von Akteuren und Themen in der Medienberichterstattung weit verbreitet (siehe z.B: Wüest et al. 2011; Leskovec et al. 2009; Miller und Riechert 2001) und können die Daten liefern, welche für die Beschreibung der Fokussierung der Berichterstattung benötigt werden. Damit können Aufmerksamkeitsschübe mit einem minimalen Aufwand auch über mehrere Jahre einer Debatte klassifiziert werden.

Mit der in diesem Beitrag vorgestellten Sekundäranalyse von Inhaltsanalysedaten und der Bestimmung medialer Aufmerksamkeitsschübe über die Abfolge von Spitzen in Volumen und Fokussierung der Berichterstattung wurde ein weiterer, sehr einfacher methodischer Ansatz präsentiert. Dieser Ansatz ermöglicht es, Daten aus bereits untersuchten und dokumentierten Inhaltsanalysen unter einem neuen Gesichtspunkt quantitativ zu betrachten und damit bisherige Interpretationen zu stützen oder zu erweitern.

Mit der Identifikation von Unterschieden im Verlauf der Fokussierung der Berichterstattung im Verlauf medialer Aufmerksamkeitsschübe legt der vorliegende Beitrag also das Fundament für eine Reihe methodischer Zugänge zur Analyse erhöhter Medienaufmerksamkeit. Dies ermöglicht einerseits eine effizientere Analyse öffentlicher Debatten, andererseits lassen sich damit qualitative Analysen zur Ermittlung der Urheberschaft von Aufmerksamkeitsschüben empirisch untermauern.

Literatur

Barabási, A.-L. (2005). The origin of bursts and heavy tails in human dynamics. *Nature, 435*(7039), 207-211.

Boorstin, D. J. (1961). *The image. A guide to pseudo-events in America*. New York: Atheneum.

Boydstun, A. E. (2008). *How Policy Issues become Front Page News*. Ann Arbor: ProQuest.

Brosius, H.-B & Eps, P. (1995). Prototyping through Key Events. News Selection in the Case of Violence against Aliens and Asylum Seekers in Germany. *European Journal of Communication, 10*(3), 391-412.

Brossard, D., Shanahan, J., & McComas, K. (2004). Are Issue-Cycles Culturally Connected? A Comparison of French and American Coverage of Global Climate Change. *Mass Communication & Society, 7*(3), 359-377.

BVG (2010). *Leitsätze zum Urteil des Ersten Senats vom 9. Februar 2010*. Bundesverfassungsgericht Deutschland, ECLI:DE:BVerfG:2010:ls20100209.1bvl000109.

Djerf-Pierre, M. (2012). When attention drives attention. Issue dynamics in environmental news reporting over five decades. *European Journal of Communication, 27*(3), 291-304.

Downs, A. (1972). Up and Down with Ecology. The "Issue-Attention Cycle". *Public Interest, 28,* 38-50.

Esslinger, D. (2010, 28. Oktober). Eine zwei vor dem Komma! *Süddeutsche Zeitung*, S. 4.

Fowler, E. F., Gollust, S. E., Dempsey, A. F., Lantz, P. M., & Ubel, P. A. (2012). Issue Emergence, Evolution of Controversy, and Implications for Competitive Framing. The Case of the HPV Vaccine. *The International Journal of Press/Politics, 17*(2), 169-189.

Imhof, K., & Eisenegger, M. (1999). Politische Öffentlichkeit als Inszenierung. In P. Szyszka (Hrsg.), *Öffentlichkeit. Diskurs zu einem Schlüsselbegriff der Organisationskommunikation* (S. 195-218). Wiesbaden: VS Verlag für Sozialwissenschaften.

International Monetary Fund (2009). *World Economic Outlook April 09. Crisis and Recovery*. http://www.imf.org/external/pubs/ft/weo/2009/01/pdf/text.pdf. Zugegriffen: 20. April 2005.

Jarren, O., & Donges, P. (2002). *Politische Kommunikation in der Mediengesellschaft. Eine Einführung*. Wiesbaden: VS Verlag für Sozialwissenschaften.

Keller, R. (2003). Distanziertes Mitleiden. Katastrophische Ereignisse, Massenmedien und kulturelle Transformation. *Berliner Journal für Soziologie, 13*(3), 395-414.

Kepplinger, H. M. (2001). Der Ereignisbegriff in der Publizistikwissenschaft. *Publizistik*, *46*(2), 117-139.

Kepplinger, H. M. (2011). *Realitätskonstruktionen*. *Theorie und Praxis öffentlicher Kommunikation: Band 5*. Wiesbaden: VS Verlag für Sozialwissenschaften.

Kepplinger, H. M., & Habermeier, J. (1995). The Impact of Key Events on the Presentation of Reality. *European Journal of Communication, 10*(3), 371-390.

Leskovec, J., Backstrom, L., & Kleinberg, J. (2009). Meme-Tracking and the Dynamics of the News Cycle. In J. Elder (Hrsg.), *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (S. 497-506). New York: ACM.

Lombard, M., Snyder-Duch, J., & Bracken, C. C. (2002). Content Analysis in Mass Communication. Assessment and Reporting of Intercoder Reliability. *Human Communication Research, 28*(4), 587-604.

Mathes, R., & Pfetsch, B. (1991). The Role of the Alternative Press in the Agenda-Building Process. Spill-over Effects and Media Opinion Leadership. *European Journal of Communication, 6*(1), 33-62.

Matthes, J., & Kohring, M. (2008). The Content Analysis of Media Frames. Toward Improving Reliability and Validity. *Journal of Communication, 58*(2), 258-279.

Mayring, P. (2007). *Qualitative Inhaltsanalyse: Grundlagen und Techniken* (9. Auflage). *UTB für Wissenschaft Pädagogik: Band 8229*. Weinheim: Beltz.

Miller, M. M., & Riechert, B. P. (2001). The Spiral of Opportunity and Frame Resonance. Mapping the Issue Cycle in News and Public Discourse. In S. D. Reese, A. E. Grant, O. H. Gandy (Hrsg.), *Framing Public Life. Perspectives on Media and Our Understanding of the Social World* (S. 106-121). Mahwah: Erlbaum.

Nisbet, M. C., & Huge, M. (2006). Attention Cycles and Frames in the Plant Biotechnology Debate. Managing Power Participation through the Press/Policy Connection. *The Harvard International Journal of Press/Politics, 11*(2), 3–40.

Pfetsch, B. (2003). Die Beobachtung und Beeinflussung öffentlicher Meinung. In B. Pfetsch (Hrsg.), *Politische Kommunikationskultur. Politische Sprecher und Journalisten in der Bundesrepublik und den USA im Vergleich* (S. 188-245). Wiesbaden: VS Verlag für Sozialwissenschaften.

Rogers, E. M., & Dearing, J. W. (1988). Agenda Setting Research. Where has it been, where is it going? *Communication Yearbook, 11,* 555-594.

Schäfer, M. S., Ivanova, A., & Schmidt, A. (2011). Globaler Klimawandel, globale Öffentlichkeit? Medienaufmerksamkeit für den Klimawandel in 23 Ländern. *Studies in Communication/Media*, o. Jg.(1), 131-148.

Schiffer, A. J. (2006). Blogswarms and Press Norms. News Coverage of the Downing Street Memo Controversy. *Journalism & Mass Communication Quarterly, 83*(3), 494-510.

Schwab-Trapp, M. (2003). Methodische Aspekte der Diskursanalyse. Problem der Analyse diskursiver Auseinandersetzungen am Beispiel der deutschen Diskussion über den Kosovokrieg. In R. Keller (Hrsg.), *Handbuch Sozialwissenschaftliche Diskursanalyse* (3. Auflage, S. 171-197). Wiesbaden: VS Verlag für Sozialwissenschaften.

Scott, W. A. (1955). Reliability of Content Analysis. The Case of Nominal Scale Coding. *Public Opinion Quarterly, 19*(3), 323-325.

Tan, Y., & Weaver, D. H. (2007). Agenda-Setting Effects among the Media, the Public, and Congress, 1946-2004. *Journalism & Mass Communication Quarterly, 84*(4), 729-744.

Vasterman, P. L. (2005). Media-Hype. Self-Reinforcing News Waves, Journalistic Standards and the Construction of Social Problems. *European Journal of Communication, 20*(4), 508-530.

Waldherr, A. (2014). Emergence of News Waves. A Social Simulation Approach. *Journal of Communication, 64*(5), 852-873.

Wüest, B., Clematide, S., Bünzli, A., Laupper, D., & Frey, T. (2011). Electoral Campaigns and Relation Mining. Extracting Semantic Network Data from Newspaper Articles. *Journal of Information Technology & Politics, 8*(4), 444-463.

Zhu, J.-H. (1992). Issue Competition and Attention Distraction. *Journalism Quarterly, 69*(4), 825-836.

Anhang A4: Identifying Clusters amid Noise

Neueinreichung nach Überarbeitung als:

Wettstein, M. (Submitted). Identifying Clusters amid Noise: Hierarchical Exploratory Cluster Analysis with Term Exclusion. Submitted to: *Communication Methods and Measures*,

Identifying Clusters amid Noise:

Hierarchical Exploratory Cluster Analysis with Noise Exclusion

Abstract

In manual and automated content analyses assessing the distribution of semantic elements across texts, multivariate analysis techniques may be used to identify recurring patterns of co-occurring elements. In communication science, cluster analysis and factor analysis are the techniques that are most often used to accomplish this task. However, these techniques are associated with two problems. First, the reliability of these methods is questionable in the presence of strongly skewed and zero-inflated count data. Second, many items that are not part of any pattern add noise to the analysis, and excluding such items in exploratory analyses can be a tedious process.

In this paper, I propose a novel method for hierarchical cluster analysis that overcomes these problems. Automated data preprocessing and the automated exclusion of any random noise present in the data allow for the reliable and replicable identification of patterns within count data matrices. Three individual studies are used to test the reliability of the method and the validity of the results. Hierarchical exploratory cluster analysis with noise exclusion is shown to differentiate reliably between randomly distributed items and patterns of co-occurrence.

Keywords: Cluster Analysis; Content Analysis; Term-Document Matrix; Count Data

For the quantitative analysis and of large matrices of count data cluster analysis and other multivariate analysis techniques for classification have been proposed and widely used to identify recurring patterns of co-occurring elements (cf. Murray, 2000; Landmann, Züll, 2004). In communication science, the main source for these count data matrices (CDM) are manual and automated content analyses where text features, such as keywords (Miller & Riechert, 2001; Hogenraad, Mckenzie, & Péladeau, 2003), semantic and formal elements (Kohring & Matthes, 2002), or issue and actor occurrences (Kriesi et al. 2006) are identified and counted across documents and represented in term-document matrices. In these cases, patterns of frequently co-occurring semantic elements are interpreted as manifest expressions of latent categories, such as frames (Kohring & Matthes, 2002) or topics (Hogenraad et al., 2003). Other sources of this kind of frequency data includes eye-tracking records and spatial or temporal event analyses which may also be analyzed for recurring patterns.

In all cases, the occurrence of elements (i.e. semantic elements, fixations, events) across a defined set of units (i.e. documents, spatial, or temporal units) may be represented in a CDM and assessed by means of multivariate analysis techniques. These matrices, however, pose two challenges for statistical analysis (Manning & Schütze, 1999). First, the elements in these matrices are unequally distributed as few elements may occur very frequently and may contain different sources of noise (e.g., infrequent or randomly appearing elements). Second, the cells of these matrices contain skewed count data with a high proportion of empty cells, which may hamper the performance of techniques operating under a Gaussian assumption such as factor analysis (Anderson, 1963). While these properties of CDMs do not prohibit the use of multivariate analyses, they require thorough manual preprocessing of the data and careful interpretation of the results.

In this paper, I propose a method for exploratory cluster analysis that is specifically designed for the analysis of sparse matrices representing count data. This method uses a two-step procedure to first automatically preprocess the CDM, and then to identify homogeneous clusters within the data while discarding noise from the solution. By excluding elements that do not belong to any homogeneous cluster from the cluster analysis, this method enables the researcher to perform exploratory cluster analyses without the preliminary identification of relevant or randomly distributed elements and outliers.

The method proposed in this paper is universally applicable to a wide range of research concerning recurring patterns of co-occurrence in frequency data from various sources. This paper, however, focuses primarily on data from manual and automated content analysis which is the major source of frequency data in communication science. Therefore, content analysis data from different

sources is used assess the performance and reliability of the hierarchical exploratory cluster analysis with noise exclusion (HECANE) under critical conditions.

Co-occurrence analyses in communication science

In communication science, count data is mainly investigated in contingency analyses of manual and automated content analysis data. Other than in relational content analyses or manual annotation of semantic relationships, these analyses search for patterns of co-occurrence in semantic or formal features of texts (Krippendorff, 2004: 203f). In order to perform this kind of analysis, features of texts are either manually or automatically identified, counted, and categorized. These features may be either formal text categories or semantic elements such as words, entities, issues, arguments, or relations among objects or actors (cf. Krippendorff, 2004: 189ff). The aim of contingency analysis is to identify meaningful, recurring patterns of co-occurrence of these features, which may be interpreted as manifestations of latent ideas in the texts.

In the manual analysis of communication events such as elections, public debates, or campaigns, contingency analysis may be used to find patterns in the data obtained through manual coding. In this analytic approach, trained coders identify the presence of a large number of individual features in the text, the co-occurrence of which may then be analyzed statistically. The features used in this analytic paradigm of text analysis may be formal criteria of the texts (e.g., Marten, 2014), specific features of journalistic presentation or framing (e.g., Semetko & Valkenburg, 2000), or constituents of interpretative frames of issues (e.g., Matthes & Kohring, 2008). This paradigm, which does not require the coders to interpret the text as a whole, has been found to yield valid and reliable results while being relatively easy to teach to coders (David, Atun, & La Viña, 2010).

In automated content analysis, contingency analysis is mainly used for unsupervised machine learning (UML; Manning & Schütze, 1999). Here, the occurrence of single words, word stems, word meanings (cf. Li, Chung, & Holt, 2008), or n-grams of words is recognized by computer programs and automatically stored in a term-document matrix (TDM). Analyses of the co-occurrence patterns of these terms may then be used to identify different types of documents within a corpus (e.g., Xu, Liu, & Gong, July 2003), issues or topics within a debate (e.g., Hogenraad et al., 2003; Quinn, Monroe, Colaresi, Crespin, & Radev, 2010), the presentation and positions of different actors (e.g., Miller, Andsager, & Riechert, 1998; Laver, Benoit, & Garry, 2003), or frames of interpretation (e.g., Veltri & Suerdem, 2013; Simon, 2001).

An alternative perspective on the analysis of TDM has been offered by document clustering techniques (for a review, see: Aggarwal & Zhai, 2012). These algorithms use the occurrence of terms in documents as proxies for their association with latent topics (Blei, Ng & Jordan, 2003) or factors (Hofmann, 1999). As both terms and documents are associated with topics in the process, an analysis

of the most strongly associated terms for each topic allows for an interpretation of its content. These models, however, do not offer any information on the strength of association between the terms within a given topic and the homogeneity of topics. Furthermore, they are specifically optimized for the analysis of natural language and require manual preprocessing (e.g. stop word removal) which limits their applicability to CDMs from different sources.

Applicability of multivariate analysis on content analysis data

A wide range of statistical methods has been used in the past for investigating the co-occurrence of semantic and formal elements, to identify recurring patterns in content data. In most cases, classification techniques such as exploratory factor analysis (EFA; e.g., Crawley, 2007), cluster analysis (CA; e.g., Matthes & Kohring, 2008), or smallest space analysis (SSA; e.g., Miller & Riechert, 2001) are used or suggested for this task. These methods have proven successful in reducing the complexity of datasets containing metric and dichotomous data. However, the application of these methods to count data matrices may lead to problems in the calculation and interpretation of results.

The standard input format for multivariate analyses in text analysis is a TDM containing information on how many times a given semantic or formal element (i.e., *term*) appears within a given textual unit (i.e., *document*). A *term* can refer to any feature of the text, and a *document* can apply to any conceivable unit of text, from a single sentence to weeks of news coverage. The size of the TDM, as well as the range of counts and the sparsity (i.e., the proportion of empty cells), depend strongly on the kind of terms and the size of the document or documents it represents.

In the analysis of TDMs using multivariate analyses, two major challenges arise from the structure of these matrices. First, as each matrix represents count data, the values within rows and columns exhibit a Poisson distribution, and may be zero in many cases. This kind of zero-inflated count data may bias the correlational measures (c.f. Huson, 2007) and spatial distances (c.f. Aherne, Thacker, & Rockett, 1998) that underlie most classification techniques. Second, the TDM may contain noise in the form of randomly distributed terms. This partial randomness not only complicates the interpretation of the solution, but also stands in the way of finding clear-cut structures.

Both of these problems associated with the application of multivariate analyses to count data have been identified and addressed before. Below, these problems and current approaches to solving them are illustrated, before introducing the approaches chosen in HECANE.

Problem 1: Applicability of multivariate analysis to sparse count data matrices

The first challenge arising in the analysis of TDM using CA or factor analysis lies in the level of the variables entered into the model. As the occurrence of terms within documents is represented by

count data, and zero is the mode for many variables, the usage of measures of concordance designed for normally distributed data may be biased.

For correlations, which are the standard measure of similarity in exploratory factor analysis, a bias toward lower correlations has been found for zero-inflated count data (c.f. Huson, 2007). As this bias increases with the overdispersion of variables, correlational structures may be misleading for this kind of data. At the same time, Pearson's correlation has been found to overweigh outliers in strongly skewed data (cf. Hauke & Kossowski, 2011). For factor analyses of TDMs, this means that high cell counts are weighed more strongly and that the most frequently occurring words and the largest documents will define the factor structure.

For squared Euclidian distance (SED), which is used in most CA methods and in smallest space analysis the converse effect is to be expected when analyzing count data. Here, the distance among very infrequent elements is bound to be small, as their positions are close to absolute zero. The distance among elements with high cell counts, on the other hand, increases disproportionately with rising deviation. For cluster analyses of TDMs, this means that all infrequent terms tend to cluster together, while all frequent terms are considered outliers.

Several approaches have been proposed to solve the problem of strongly skewed data in multivariate analysis. First, logarithmic or square root transformations of the data can decrease skewness and reduce the number of outliers exhibiting large counts (cf. Sirkin, 2006). Second, the cell counts may be weighed by the entropy of terms (cf. Landauer & Dumais, 1997) or by the term- and document frequency (TF-IDF weighting: Salton & Buckley, 1988). Third, the measure of similarity or distance may be adapted to the data structure by using non-parametric correlation (cf. Zegers & Ten Berge, 1985), or by correcting chi-square for the total sample size (cf. Aldenderfer & Blashfield, 2006). Fourth, if available, information on the kind of association between terms and documents may be used to compute metric coefficients instead of count data (cf. Kriesi et al., 2006). Fifth, the matrix may be transformed into dichotomous data (cf. Matthes & Kohring, 2008) to eliminate all outliers and reduce the problem of varying distances. Finally, the rows and columns of the matrix may be standardized or normalized to unit length prior to the analysis to attain the same scale for all variables and balance the weight of frequent and infrequent elements.

Problem 2: Sources of noise in TDMs

A second problem in TDMs representing natural language may arise from noise that is present in the documents under investigation. Specifically, there are three kinds of terms that have the potential to blur any patterns of co-occurrence present in the data. First, there are terms that are present in most documents. These may be stop words – such as pronouns and ubiquitous prepositions – when using single words, or main lines of argument when using broader semantic

units. These terms are likely to co-occur with most other terms and add meaningless content to the eventually found patterns. While the removal of stop words is simple in languages with known stop word lists when using single words, their detection in analyses using n-grams (c.f. Fürnkranz, 1998 / Rlioff, 1995) and when using formal or broader semantic elements may be challenging.

Second, in addition to these very frequently occurring elements, there is also likely to be a large group of terms that do not occur in most texts and therefore score a total count of zero in most cells. In spite of these terms probably being constituents of patterns, they rarely co-occur with most words and may therefore be mistaken for outliers.

Finally, there is likely to be a group of terms that are randomly distributed across all texts without being frequent. These terms are not associated with any other terms and therefore add to the randomness of the matrix and impede the determination of the optimal number of clusters or factors to extract in the analysis.

All three kinds of noise lead to partial randomness in TDMs and blur patterns of co-occurrence that are actually present. In content analyses, this is a severe problem because the patterns to be expected in the co-occurrence of semantic elements are not as clear-cut as those of strongly correlated items from questionnaires, which are usually investigated using multivariate classification techniques. In most cases, this noise increases the tendency of scree plots and elbow plots to become straight curves, providing no valuable information on the number of dimensions or clusters to be expected.

To reduce noise originating from very frequent and infrequent terms, upper and lower bounds of frequency may be defined by the researcher (cf. Matthes & Kohring, 2008). In addition, when analyzing the co-occurrence of single words, a predefined list of stop words may be used to exclude certain randomly distributed words from the analysis. In cases where no stop word list is available, the calculation of term entropies may be alternatively be used to determine these terms (Aggarwal & Zhai, 2012: 83).

A greater challenge is posed by moderately frequent words that are randomly distributed across documents and that do not contribute to the solution. These terms do not belong to any single pattern and have to be excluded because of their weak association with the solution. In EFA, these terms may be removed by eliminating terms with low communalities and high double loadings. In CA and SSA, they may be excluded when they are detected as outliers. In both cases, however, the identification of these terms is a tedious process demanding a series of potentially consequential decisions by the researcher.

Another way to deal with all three sources of noise is to define a set of keywords to be used in the analysis. By manually narrowing down the data to a select sample of relevant terms, any noise originating from randomly distributed, meaningless terms is reduced. The selection of a set of keywords, however, may, again, be a tedious process, and may include arbitrary decisions made by the researcher.

Proposed clustering technique for count data

In this paper, I propose a method for cluster analysis that may overcome the problems of data structure and noise without requiring manual preprocessing and term selection. This method was expressly designed for finding patterns of co-occurrence in TDMs and is therefore adapted to zero-inflated count data and the level of randomness that is expected in data representing media content. It is, however, universally applicable to any kind of matrix representing count data and may also be used to find patterns of co-occurring words in natural language or events in event data.

The method is performed in two steps, the first automatically preprocessing the data and the second identifying relatively homogeneous clusters of terms, automatically excluding all terms that do not belong to any of these clusters. The manual steps required in exploratory factor and cluster analyses are thereby replaced by a replicable and sensible automated process. Hierarchical exploratory cluster analysis with noise exclusion (HECANE) is therefore a suitable method for finding small groups of frequently co-occurring elements in large matrices of count data with a high likelihood of noise and unequally distributed elements.

Step 1: Data preparation and distance metric

The first step in the HECANE process is the preprocessing of the CDM to render it suitable for cluster analysis. The net frequencies of elements are likely to differ strongly in count data originating from content analysis or event data. As I seek to use well-established measures of distance in the ensuing analysis, the matrix needs to be transformed to balance the weight of frequent and infrequent elements.

This transformation has to fulfill two main requirements to obtain reliable and informative measures of distance during CA. First, for elements with different net frequencies but similar distributions, the values should be similar in the transformed matrix. This means that the similarity of two elements should be highest when the relations among occurrences in different documents are equal, regardless of their absolute counts. Second, for elements with no common occurrence in any group, distance should be maximal. Again, this should hold for both frequent and infrequent elements.

To meet both requirements, I chose to use a normalization of the columns of the matrix. Thus, the vectors of all elements are scaled to a length of one, and the relative distribution of each element is maintained while the global abundance of items is neglected (cf. Podani, 2000: 42). Furthermore, all values of zero in the original matrix remain zeroes, which would not be granted by standardization.

For any *N*x*M* CDM (*C*) with *N* elements in *M* groups, the normalization $C \rightarrow C^n$ is defined as follows:

$$\forall(i \in N): \vec{c}_i^n = \frac{\vec{c}_i}{\sqrt{\sum_j^M (c_{ij} - \bar{c}_i)^2}}$$

The result of this normalization is a transformed matrix where all elements are geometrically represented by points on the surface of the non-negative segment of an *N*-dimensional sphere with a radius of 1. On this sphere, elements without any common occurrence lie on different sides of the rim, with a total SED of 2, which is the greatest possible distance. Conversely, elements with similar distributions are close together, regardless of their absolute count. Likewise, the angle among elements with similar distributions is small, while elements without co-occurrence are orthogonal and therefore have a Pearson's correlation of 0. The requirements specified above are therefore met for the standard measures of similarity and dissimilarity both in cluster analysis and factor analysis.

As all points lie on the non-negative segment of the surface of a sphere, Pearson's correlation, angle, and SED all yield the same ranking of distances. The choice of measure of similarity is therefore of no consequence for the result. However, as hierarchical cluster analyses with large numbers of elements require many steps, in each of which all distances need to be calculated, I propose the use of SED. Of the three possibilities, this measure of distance may be computed most efficiently. The distance between two columns (c_1 and c_2) in the matrix is defined as the sum of squared deviances over all *M* rows:

$$D(c_1, c_2) = \sum_j^M (c_{1j} - c_{2j})^2$$

Step 2: Cluster analysis routine

The second step in the HECANE process is a bottom-up hierarchical cluster analysis linking cluster gravity centers (c.f. Manning & Schütze, 1999; Aldenderfer & Blashfield, 2006). This means that the procedure successively couples the elements with the lowest distance until one single hierarchical cluster containing all elements is reached. As reasoned above, SED is used as a measure of distance because of its efficiency, even in the case of large datasets.

Unlike other CA methods, the routine then employs an innovative post-processing of the solution, whereby all relatively homogeneous clusters are extracted from the hierarchical tree, discarding the rest of the solution. The method thus results not in a single tree but in multiple homogeneous clusters representing only a part of the elements entered in the analysis. Any elements not included in a homogeneous cluster are considered noise and are not considered part of the solution.

Decisions about which clusters and elements are to be selected and which are to be discarded are reached in the process of hierarchical cluster analysis by applying a single rule: If the inclusion of the nearest element (*e*) to an existing cluster with a gravity center (*g*) will lead to a new cluster with a radius exceeding the distance between *g* and *e*, the element is not to be included, and the cluster is to be considered complete. That is to say, if the inclusion of a new element to an existing cluster would result in a new gravity center closer to the new element than to the already incorporated elements, this inclusion is considered detrimental to the overall solution (see Figure 1). This rule prevents both the incorporation of extreme outliers and the fusion of distinct and homogeneous clusters.

<<Figure 1 about here>>

Implementation and output

For this initial paper on the method, a basic Python implementation of HECANE is used. The script requires less than 200 lines and may be used to perform both steps of the method on any matrix of count data. A summary of the results is then printed in plain text. As the routine is quite easy to implement, the inclusion of the method in existing statistical packages is possible, but has not yet been released.

Unlike other CA methods, the result of HECANE is not a single dendrogram containing all elements submitted to the analysis, but rather a series of small dendrograms representing the homogeneous clusters within the data. Any elements that do not belong to these clusters are not part of the results and are not further evaluated.

The output of the routine should therefore mainly include a list of clusters and the elements that each cluster contains. The inclusion of additional information, such as the diameter of the clusters, the distances between their gravity centers, and the absolute frequency of each element belonging to a cluster, is recommended. This information may help in the interpretation and evaluation of the results.

In addition to this primary output providing an overview of the solution, the cluster gravity centers may be exported and attached to the initial count data matrix. This enables the researcher to conduct subsequent analyses, such as a typology of documents, based on the results of the cluster

analysis. Graphical output, such as dendrograms for each cluster and spatial visualizations of the proximity of terms and clusters, may also be implemented.

Reliability and validity

The method proposed in this paper is a tool for the measurement of the existence and content of patterns of co-occurrence in count data matrices. As with all measures, the interpretation and evaluation of HECANE's results strongly depend on its reliability and validity.

Reliability, in this context, concerns whether the method measures the desired property of an object and whether the results are robust in repeated measurements of the same object. With regard to the cluster analysis of frequency data, a distinction between discriminative reliability and robustness may be made. *Discriminative reliability*, in this respect, refers to the ability of the method to distinguish between actual patterns and noise. For this method, specifically, the exclusion of randomly distributed items from the solution while retaining meaningful patterns is regarded key to the reliability of the measurement. *Robustness*, on the other hand, refers to the stability of the solution in the presence of new information. The results should change only marginally with extensions of the CDM by additional elements (e.g. terms) or groups (e.g. documents).

Validity concerns the representation of reality obtained through a measurement. Here, I differentiate between two types of validity that are important in the statistical analysis of texts. First, I seek to establish *semantic validity*, which is centered on the question of whether the patterns described in the solution are meaningful, coherent, and interpretable in the context of the object under investigation (cf. Quinn et al., 2010). Second, the results have to be *externally valid*, meaning that the patterns found are in agreement with results from other assessments of the same object. This includes both the formal agreement and the plausibility (i.e., face validity) of the results.

For the purpose of a critical test of HECANE with regard to these quality requirements, three studies are conducted on three different types of CDM that may pose problems to other multivariate classification techniques. Specifically, I aim to find patterns of co-occurrence among elements in very unevenly distributed data, natural language, and randomly distributed data. The aim of these studies is a critical discussion and evaluation of the reliability of the proposed method and the validity of its obtained results.

Study 1: Clustering of unequally distributed terms

In this first study, data from a manual content analysis are used to determine common frames of reference within a debate. As Matthes and Kohring (2008) have pointed out, one reliable way of finding common frames is to manually identify constituents of frames and then use cluster analysis or factor analysis to find common patterns. In the past, this method of identifying frames has proven

successful in different debates and with different sets of frame constituents taken into account (cf.Matthes, 2007; David et al., 2010). Frame constituents may be problem definitions, causal attributions, moral evaluations, or courses of action, following Entman's (1993) seminal review on framing.

Using cluster analysis to detect patterns of co-occurrence, however, may prove difficult due to unequal distribution and possible random elements. For example, it has been found that the number of clusters is hard to determine because the elbow criterion may point to multiple possible solutions in the presence of randomly distributed items. The decision regarding the optimal number of clusters is therefore likely to be backed by the interpretability of cluster solutions, rather than by the elbow criterion (cf. Matthes & Kohring, 2008, p. 269).

Furthermore, as Ward's method and k-means cluster analyses use the SED as a measure of similarity, the data have to be preprocessed manually to compute the analysis. Matthes and Kohring (2008) propose transforming the count data matrix into a dichotomous matrix, with each cell indicating only whether a term was present at all in a given document. To prevent infrequent elements from clustering together, they suggest removing all terms that occur in less than 5% of the documents. These steps, while making cluster analysis of count data feasible, result in a massive loss of information on the distribution of terms and on the occurrence of rare terms.

The present study applies the paradigm proposed by Matthes and Kohring (2008), using HECANE to identify common frames of reference in the debate on unemployment in Great Britain before and after the announcement of the cut of half a million public sector jobs in October 2010. This announcement, made by the British Chancellor George Osborne changed the focus of the debate on unemployment from a problem of welfare and budget to that of individuals in public service who were put out of work and needed support and new jobs (cf. Daily Hansard, 2010).

The aim of this first study is to establish the reliability and external validity of the results by means of a detailed analysis of the output. To be considered a reliable method for finding patterns of co-occurrence of semantic elements, the routine should exclude randomly distributed and infrequent terms. The solution should mainly consist of moderately frequent terms with non-random distributions across the documents. Validity may be established by means of a critical comparison of the results with previous findings on the same issue, as proposed by David et al. (2010). Here, the shift of the political debate sketched above shall be used as a litmus test for the external validity of results obtained by HECANE.

Method

To find frame constituents within the Fall 2010 debate on unemployment, a manual content analysis was performed on 2232 texts from seven major British newspapers: *The Guardian, The Daily*

Mirror, The Daily Mail, The Telegraph, The Sun, News of the World, and *The Observer*. The sample was divided into texts published before (Sept. 10 to Oct. 14; *N* = 644) and after (Oct. 30 to Dec. 10; *N* = 760) with the announcement of job cuts made by Chancellor Osborne on October 19. For each text, any statement on a specific problem definition, cause, effect, or treatment of unemployment was identified and coded. The coding scheme used in this analysis contained 118 different instances of such frame constituents. Intercoder reliability among the nine coders in the project was assessed using a hidden reliability test in which 88 texts were distributed to all coders as a part of their usual work throughout the analysis. Because of the hidden nature of the test and the fine-grained assessment of frame constituents, the intercoder reliability was low but acceptable, with a Scott's pi of 0.573 (Scott, 1955; Lombard, Snyder-Duch, & Bracken, 2002).

To find common frames of reference, the data were transformed into a TDM on the level of individual speakers, with all statements made by an individual speaker treated as a document and each frame constituent treated as a unique term. For the phase before the announcement of job cuts (Phase 1), the matrix contained 88 individual speakers and 66 different frame constituents. In the period after the announcement (Phase 2), 78 individual speakers were identified using 77 different frame constituents. The skewness of the matrix was computed using the Gini coefficient for distribution inequality (see Atkinson, 1970).

The number of statements on frame constituents was unequally distributed among speakers, with 90% of the statements made by 44% of the speakers (Gini coefficient = 0.702) in the first phase and by 32% of the speakers in the second phase (Gini coefficient = 0.760).Likewise, the frame constituents were distributed unequally in the two phases, with the 10 and 12 most frequent terms making up 50% of the total number of terms in Phases 1 and 2, respectively (Phase 1: Gini coefficient = 0.513; Phase 2: Gini coefficient = 0.497). In addition to this uneven distribution, the matrix was found to be sparse, with 86.9% of the cells in the first phase and 86.1% of the cells in the second phase having a value of zero.

Data analysis using HECANE resulted in an interpretable solution for the debate in Phase 1. Four homogeneous clusters were found using a total of 25 frame constituents. The remaining 41 frame constituents were considered noise by the routine and automatically excluded from the solution. A detailed look into the terms excluded from the solution revealed that they included both the most frequent terms of the analysis, which appeared in more than 45% of the documents, and many of the most infrequent terms, which appeared in less than 10% of the documents.

Upon closer investigation, it became apparent that some infrequent terms were included in the analysis while other moderately frequent terms were excluded. A possible explanation for this behavior of the routine may lie in the actual randomness of the distribution of individual terms. This

explanation is supported by a binary logistic regression predicting the inclusion of items depending on their frequency and the entropy of their distribution (cf. Hellman, 2001). The results indicate that terms with a higher frequency (B = 1.43; SE = 0.61; p < .05) and lower entropy (B = -11.85; SE = 4.96; p < .05) had a better chance of being included. Conversely, the less frequent and more randomly distributed terms were excluded as noise.

The four clusters identified by this method may be interpreted as actual frames of reference that were used by certain speakers and avoided by others in the media coverage that was investigated. These frames of reference thus indicate the different interpretations of unemployment in Great Britain before the announcement of massive job cuts in the public sector. In Table 1, the different frames of reference are summarized by indicating the frame constituents they contain. Accordingly, the allocation of a cause, problem definition, and effect in one frame of reference is interpreted as a frame emphasizing a causal link between these elements.

<<Tables 1 and 2 about here>>

The first frame of reference was centered on unemployment on a global scale, mostly outside the UK, which was caused by business closures and which might have had an impact on domestic politics. Second, domestic employment rates and youth unemployment caused by labor market structures were debated, with an increase in financial support and a reform of unemployment benefits discussed as possible solutions. Third, a set of possible measures to reduce the unemployment that was seen as a result of the generous welfare system was debated. These measures chiefly included a redistribution of the budget, reducing financial support for unemployed persons, and investing in education and job-seeking assistance. Finally, there was debate on specific cases of individuals losing their jobs because of business fluctuations and mass redundancies. Here, individuals' fear and psychic distress were discussed as the consequences of unemployment. In sum, in the first phase, the overall unemployment debate was divided into a debate on foreign unemployment, youth unemployment, welfare abuse, and the individual consequences of job loss.

To assess the external validity of the results, Ward's method of cluster analysis was applied to the same data after vector normalization to confirm the existence of these four clusters. Because the research interest in this study concerns the patterns of co-occurrence of items and no information on the kind of association between actors and frame constituents is known, a cluster analysis using Ward's method is the most suitable choice. As may be seen in Figure 2, an analysis of the normalized count data matrix results in an elbow plot with several weak elbows, indicating 3, 9, 13, or above 20 possible clusters. Subsequent removal of outliers would have to be conducted to achieve a result pointing clearly toward an optimal number of clusters. When only the 25 frame constituents that remained after the second step of HECANE were used, the elbow plot clearly points toward the same

four clusters discussed in the preceding paragraph. These findings indicate that the same clusters may be found using Ward's method of cluster analysis when random noise is removed from the TDM prior to the analysis, or by means of manual exploratory cluster analysis.

<<Figure 2 about here>>

For the second phase, the same analysis was performed, resulting in seven clusters accounting for 41 frame constituents. A total of 36 frame constituents were again excluded as noise for the same reasons as in the first phase. As expected, the frames of reference represented by these clusters were not only more differentiated, but also slightly different from the ones observed before the announcement of public sector job cuts (cf. Table 2).

First, there was again a debate on unemployment abroad and from a global perspective. Second, the depiction of specific cases of unemployment was now not only linked to business restructuring, but also to budget cuts and to a decreasing number of jobs. Third, unemployment in general was attributed to the financial crisis, and political consequences were discussed. Fourth, only long-term unemployment was now discussed as a consequence of a generous benefits system—again, the restriction of financial support was proposed as a solution. Fifth, a set of measures was discussed to alleviate problems resulting from structural unemployment; these measures were mainly investments in education, an increase in financial support, and increasing the number of jobs in the private sector. Sixth, the outsourcing of jobs and emigration were discussed as causes of domestic unemployment, leading to problems for both families and the national economy. Finally, there was a heterogeneous debate on the possible consequences of unemployment, ranging from emigration of the workforce to strikes and negative consequences for domestic businesses.

In conclusion, the main lines of reasoning in the debate on unemployment were changed by the announcement of public-sector job cuts. First, the role of an overly generous welfare system as a possible source of unemployment was more differentiated after the announcement. Second, budget cuts were introduced as a new major source of structural unemployment, especially in combination with specific cases. Finally, new solutions, especially supportive ones, were discussed to alleviate the problems and possible consequences of unemployment. These findings are consistent with initial expectations and with a superficial overview of media coverage.

Discussion

Using HECANE to identify recurring patterns of communication in the debate on unemployment in Great Britain, a set of recurring patterns of frame constituents were found that represent major frames of reference in the debate before and after the announcement of severe public-sector job cuts. The emerging clusters are meaningful and in accordance with expectations, which points toward the semantic and external validity of the results.

By using HECANE to identify clusters, problems that are typically experienced when submitting sparse count data to Ward's method of cluster analysis could be overcome, and noise could be removed from the data. The automated exclusion of very frequent, very infrequent, and randomly distributed terms, which would not be excluded in standard CA, indicates discriminative reliability of the measurement. While the frame constituents excluded as noise may have been important to the debate and are themselves not to be considered as meaningless noise, these elements were not part of any homogeneous pattern. Instead, they may have been located in the overlap of two patterns or used in isolated cases only. Other analysis methods, such as issue ownership analyses by means of multidimensional scaling (cf. Kriesi et al. 2006) may be applied to clarify their role in the debate.

In conclusion, the method proposed in this paper proves to be a reliable tool for obtaining valid semantic clusters from manual content analysis data. It is therefore considered a suitable statistical procedure to identify frames of reference using the analytical paradigm proposed by Matthes and Kohring (2008).

Study 2: Clustering of words in natural language

In the second study, a word co-occurrence analysis in natural language texts is applied to identify meaningful groups of frequently co-occurring words, indicating different topics. Specifically, the aim of the study is an assessment of frequently recurring speech patterns used by politicians on Twitter. While the use of multivariate analysis in the area of natural language processing has been challenged and partially replaced by elaborated document clustering techniques such as latent Dirichlet allocation (LDA: Blei et al. 2003), multivariate classification techniques may be applied in political and communication science to categorize texts, identify sub-issues within debates, and distinguish the communication frames of different actors (Crawley, 2007; Miller et al., 2008; Simon, 2001). The advantages of multivariate analysis in these cases are the requirement of only a small number of documents or groups and the focus on the correlation between elements instead of their discrimination between latent topics. Therefore, they may be applied to directly answer questions regarding patterns in aggregated frequency data.

When CA is used in the identification of pattern in natural language, k-means cluster analysis is recommended (Aggarwal & Zhai, 2012: 92) because this procedure is more effective than hierarchical CA in the presence of large numbers of elements (cf. Shih, Rennie, Chang & Karger, 2003). As this method also relies on Euclidian distances, it requires manual preprocessing of the term-document matrix (TDM) to decrease sparsity and balance the weight of frequent and infrequent words. In addition, stop words such as prepositions, pronouns, and common phrases must be excluded prior to analysis (Manning & Schütze, 1999) to reduce the noise.

As HECANE has been shown in Study 1 to distinguish randomly distributed terms from actual patterns, manual preprocessing of the TDM is not necessary when using this method. In addition, there is no need to predefine the number of clusters to be extracted, as the method will return all relatively homogeneous clusters that are present in the data. There are, however, two major differences between content analysis data and raw TDMs representing natural language. First, the share of random terms is likely to be higher in natural language because it contains many context-dependent words (e.g. prepositions and articles). Second, the number of terms is substantially larger when analyzing natural language. Most of these terms, however, are very infrequent and are unlikely to contribute to frequently recurring patterns. It is therefore recommended that infrequent terms are removed prior to the analysis for reasons of efficiency and controlling noise (Blei et al. 2003).

The aim of this second study is a critical evaluation of HECANE in the task of finding co-occurring words in natural language texts. Specifically, I test whether stop words and infrequent words are correctly identified as noise, whether the solution is robust upon the inclusion of additional infrequent terms, whether other random words are excluded from the solution, and whether the emerging clusters of words are interpretable based on their semantic relationship.

Method

The research interest of this exemplary test of HECANE in clustering natural language is the question for recurring patterns of terms in the communication of single actors betraying different styles of twitter use. To perform this analysis, an automated content analysis of political tweets in the US was performed. The analysis included 15,970 tweets issued in March and April 2014 by 11 US politicians who are very active on Twitter.

The terms used in the analysis were single words, bigrams, and trigrams occurring in the tweets (N > 370'000). Due to the high number of terms it is necessary to remove very infrequent terms as they may not contribute to recurring patterns and reduce the efficiency of hierarchical cluster analysis. To assess the sensitivity of the method to the decision on the number of terms included, three analyses were performed using different thresholds for the exclusion of infrequent elements. Terms appearing in at least 1% (N = 228), 0.5% (N = 516), and 0.25% (N = 1080) of the tweets, respectively were used in the three analyses. The documents used for the TDM were the complete feed of tweets issued by each politician (N = 11).

The TDMs contained the count of every term for each document and were not preprocessed further than excluding the infrequent terms in each analysis. Because of the large number of tweets per document (M = 1541; SD = 706), the sparsity of the matrices was relatively low, with only 2.7% of cells containing the value 0 for the matrix with words occurring in at least 1% of the tweets (compared with 7.1% for 0.5% and 14.4% for 0.25%, respectively).

Three exploratory cluster analyses with automated term exclusion of these TDMs resulted in a similar solution for each set of terms. In each cluster analysis, one large, homogeneous cluster was found, comprising 11–19 smaller clusters of stop words. Apart from these stop word clusters, a varying number of meaningful clusters was found for each set of terms (see Table 3).

An alignment of the solutions showed that additional clusters found after including more terms in the analysis were predominantly sub-clusters of previously found clusters. Specifically, in the second analysis, which used the 516 most frequent terms, 87.9% of the 228 terms from the first analysis ended up in sub-clusters of the clusters they previously formed. Only 12.1% of the terms changed to other clusters. In addition to the sub-clusters that were formed in the second analysis, 11 new clusters were found that comprised only new terms.

The third analysis, using the 1080 most frequent terms, exhibited an even higher degree of similarity to the first analysis, as 94.3% of the terms used in the first analysis were found in new sub-clusters. This indicates that some changes that resulted from the inclusion of new terms in the second analysis were amended by the inclusion of even more terms in the third analysis. The similarity to the solution from the second analysis was also high (88.2%), with 17 new clusters containing only new terms (see Table 3).

A semantic assessment of the cluster solution generated for the first analysis showed that each cluster holds an interpretable set of terms representing certain kinds of tweets. Table 4 provides the content and overarching semantic concepts of all meaningful clusters. The clusters could be categorized as functional tweets (*N* = 7) expressing appeals and information, or as issue tweets (*N* = 7) containing policy statements. In order to establish the external validity of this cluster solution, a Ward cluster analysis was computed using the same 86 terms which made up these final 14 clusters. Again , the analysis resulted in a 14-cluster solution which is highly congruent with the solution found by HECANE. Only 7 terms were relocated in a different cluster.

In conclusion, the results obtained in three different analyses using different sample sizes of terms were strikingly similar. The solution was shown not to change in response to the inclusion of additional noise and may therefore be considered robust. Furthermore, the branching behavior of the solution when additional elements are included is desirable when investigating patterns of co-occurrence in texts, as the solution becomes more differentiated with increasing information.

<<Tables 3 and 4 about here>>

Discussion

The aim of the second study was a critical evaluation of the behavior of HECANE in the face of unprocessed TDMs representing natural language. As these matrices contain different sources of

noise, they usually require extensive manual preprocessing prior to most statistical analyses. In this test, the raw matrices were used, and the resulting solutions were robust and semantically valid.

One notable result of this test is the behavior of the method in the presence of stop words. In each version of the analysis, the stop words were grouped in one major cluster containing of 11–19 sub-clusters that were clearly discernible as stop-word clusters. These clusters represent actually recurring patterns as the ratio of stop words varies with the size of documents and is therefore similar across documents with different size. The recurring similar ratio is therefore correctly recognized as a pattern resulting in a large and homogeneous cluster. This cluster may easily be excluded from the semantic interpretation of the solution. As this behavior deviates in a positive way from other CA methods, it points toward the desired discriminative reliability of HECANE.

Furthermore, while the majority of stop words was grouped in this stop word cluster, there are some words present in the patterns which would be excluded using standard stop word lists. Terms like "my", "too", "be on", or "I will" would have been excluded. However, as this analysis uses tweets, the removal of these terms would have been detrimental to the interpretation of the data. They are clearly associated with a certain type of tweets and are vital parts of recurring patterns of co-occurrence in this mode of communication. The ability of HECANE to make a difference between patterns and noise allows for these analyses to be conducted without the initial removal of stop words and therefore without the risk of losing crucial elements.

Another key finding is the reaction of the procedure to the inclusion of large numbers of infrequent words. Instead of restructuring the solution completely, additional words mainly led to the diversification of previously found clusters or formed new clusters made up of new terms. For the most part, the solution remained robust upon the inclusion of additional information.

Study 3: Clustering of random noise

The third study tests the discriminative reliability of HECANE by applying it to randomly distributed data, which should not result in any solution. While it is possible to extract any number of clusters from random data using other cluster analysis procedures, HECANE should automatically discard noise from the solution and produce solutions without any meaningful clusters.

To evaluate the solutions found in the analysis of random noise, I use random datasets resembling the unequal distribution of terms in the first study. The formal likeness of the datasets enables a direct comparison between the solutions achieved for meaningful and random data. Thus, any difference in the behavior of HECANE in the presence of meaningless data may easily be detected.

Method

For this study, a set of 100 random datasets was created by means of a Monte Carlo simulation. To enable a comparison of the results of the analysis with solutions for non-random datasets, the data were simulated to resemble the unequal distribution of the content analysis data in Study 1 as closely as possible. Each dataset contained 87 documents and 66 terms with a distribution similar to those found in Study 1. The mean Gini coefficient was .525 (*SD* = .011), which was only slightly higher than the inequality measured in the first analysis of Study 1 (Gini coefficient = .513). The sparsity of the simulated datasets (M = 78.9%; SD = 0.2%), however, was lower than that of the original study (86.1%). Despite these small deviations from the data of the first study, the simulated datasets were judged similar and submitted to HECANE. As in the first study, no preprocessing was performed.

Results

The analysis of all datasets using HECANE resulted in 100 solutions, each containing at least one cluster. Each of these solutions was assessed manually to determine the content of the resulting clusters and the absolute frequency of the constituting terms.

In 58 cases, only one cluster was found. In 31 cases, the analysis resulted in two clusters, and in 11 cases, three clusters were identified. In all solutions, the first cluster to be found consisted entirely of very frequent terms (occurring 0.25 to 2.5 times in each document) and contained an average of 22.0 (*SD* = 4.9) elements. A close investigation of these clusters revealed that they represent actual patterns created by randomly distributing terms with different absolute frequencies. As these frequencies are forced in all documents in the simulation, a pattern of similarly distributed terms emerges. This is only possible, however, for frequently occurring terms.

In cases where more than one cluster was found, the smaller clusters comprised between four and six elements, and each contained one to three very infrequent terms (occurring less than 0.1 times in each document). These clusters evidently resulted from the random co-occurrence of rare elements in single documents. When elements only occur once or twice in the entire dataset, random co-occurrence in one of these rare opportunities counts strongly toward their proximity. These clusters are therefore regarded as measurement artifacts.

In 13 solutions, one small cluster of moderately frequent terms was found. These clusters had the structural appearance of the meaningful semantic patterns found in Study 1. In no case, however, was more than one of these patterns found; in contrast, the analysis in the first study resulted in four to seven clusters of similar size consisting of moderately frequent terms.

Discussion

Contrary to initial expectations, HECANE offered a result for each of the 100 randomly simulated TDMs in this sample. These solutions, however, never contained more than one reasonable cluster that might be mistaken for a meaningful semantic pattern within the documents under investigation. Solutions with a structure comparable to that found in Study 1, containing four to seven meaningful clusters of similar size, were not found in the random data.

The solutions always contained one large cluster of very frequent words representing the pattern forced upon the data by forcing the same unequal distribution of terms on all documents. The structure of this cluster is comparable to the stop word clusters found in the analysis of natural language in Study 2. In addition to this large cluster, some smaller and potentially interpretable clusters were found in almost half of the cases. These clusters, which were mainly made up of very infrequent terms, were evidently actual patterns formed by the random co-occurrence of infrequent terms in single documents.

In conclusion, obvious structural differences are to be expected when using HECANE to cluster meaningful versus random data. The analysis of random TDMs is likely to result in exactly one cluster containing the most frequent terms. In addition, there is some chance that random actual co-occurrences of infrequent terms will be identified as patterns and returned as part of the solution. In contrast, the solution of actual content analysis data in Study 1 contained several clusters of approximately equal size, each containing both frequent and infrequent terms.

The structural difference between the solutions found in random and non-random data points toward a strong discriminative reliability. While most cluster analysis routines allow the extraction of any number of clusters from random datasets, HECANE will only return large stop-word clusters, as well as small clusters reflecting odd patterns present in the random data. These solutions are qualitatively different from the solutions found when using meaningful data, as in Studies 1 and 2.

In order not to commit a type 1 error when interpreting solutions from HECANE it is important to monitor the absolute frequency of terms included in the single clusters and the diameter of the initially found clusters. Both the stop word clusters in study 2 and the invariant first cluster in study 3 were found first and were the most homogeneous clusters in the solution. At the same time, they contained the most elements. The artificial clusters in study 3, on the other hand, were found later in the process and contained mostly infrequent terms. In the interpretation of solutions, caution is advised in cases where the first and most homogeneous cluster contains the most elements and when clusters are made up entirely from very frequent or infrequent terms. While these clusters do, in fact, represent actual patterns in the data, their content should not be interpreted as latent semantic structures but as data artifacts.

General discussion

In contingency analyses of counted elements, such as semantic elements in quantitative content analyses, cluster analysis may be used to identify recurring patterns of co-occurrence. The nature of count data of semantic elements, however, requires preprocessing prior to the analyses to render them suitable for multidimensional scaling. Furthermore, the exclusion of items not contributing to any pattern has been found to be a tedious process in exploratory analyses.

The hierarchical exploratory cluster analysis with noise exclusion (HECANE), which is introduced in this paper, may solve previous issues of the applicability of multivariate analysis to sparse count data matrices. In this method, after the automated preprocessing of the data, a novel method for hierarchical cluster analysis is applied that reliably differentiates between randomly distributed elements and recurring patterns. In the process of this analysis, all elements not contributing to homogeneous patterns are excluded from the solution. HECANE is therefore applicable to CDMs without preliminary or stepwise selection of relevant and irrelevant elements.

Three studies were performed to test the reliability of this fully automated pattern detection in different sets of sparse count data matrices. The method was found to have high discriminative reliability, as it reliably separated randomly distributed items from meaningful patterns, either by wholly excluding them from the solution or by isolating them in large and structurally unique clusters. In addition, the method has been shown to produce robust solutions that are only marginally affected by decisions on thresholds for infrequent terms to be included in the analysis.

The validity of the results was more difficult to establish than reliability in the studies included here. It could be shown, however, that all results were semantically valid, as the emerging clusters formed meaningful and interpretable semantic units. Furthermore, the change of communication patterns found in the first study is consistent with expectations and was found to be plausible in light of the associated debate. From these results, I conclude that the method proposed in this paper is a reliable way to find recurring patterns of co-occurring elements in count data. Moreover, as it does not require manual preprocessing or exploration of the data, it is a very efficient way to do so.

Concerning both the test of validity and reliability it has to be noted, that the paper only superficially compares the results of HECANE to the results of previously used procedures (e.g. Ward cluster analysis). This method was designed to substitute the tedious process of manually excluding items in exploratory factor and cluster analyses Any comparison with techniques retaining all elements in the solution would be futile. A comparison with a well-documented exploratory cluster analysis, on the other hand, would exceed the range of this paper. Likewise, while there are training sets and reference corpora for document clustering algorithms, there is no reference dataset for the identification of recurring patterns amid randomly distributed elements in frequency data. The

discussion of reliability and validity of HECANE is therefore limited to the feasible modes of assessment in this paper.

The initial analyses presented in this paper highlight two limitations of HECANE. First, while the fully automated process that removes the necessity for decisions by the researcher is useful for an efficient and quick analysis of count data, it also renders the technique unresponsive: If no pattern is found or if crucial elements are lost as noise in the process, there is no possibility to amend the solution by means of adjusting parameters, as can be done in other clustering methods. Furthermore, the solution tends to narrow the scope of possible interpretations through the dichotomous distinction of noise and meaningful elements that are unambiguously assigned to clusters.

A second limitation concerns the certainty and statistical inference of results generated using this method. Owing to the small number of documents and counts in the third study, the random co-occurrences of terms in single documents were interpreted as parts of actual patterns, resulting in random small clusters of infrequent elements. These artifacts may also occur in content analysis data utilizing small samples. To prevent mistaken interpretations of the data, clusters containing infrequent terms should be scrutinized before accepting them as actual semantic patterns. To do this, I suggest using a jackknife resampling (Efron, 1982) to check the robustness of these clusters upon the omission of single documents and terms.

In conclusion, the novel method for cluster analysis of count data proposed in this paper may prove to be a useful tool in the detection of recurring patterns of co-occurrence in different kinds of frequency data (e.g. content analysis data, event data). Future implementations of the routine in statistical packages or as a stand-alone program may include jackknife sampling or different preprocessing routines to render the method less rigid and more reliable in the distinction between measurement artifacts and meaningful patterns.

References

Aggarwal, C. C. & Zhai, C.X. (2012). A Survey of Text Clustering Algorithms. In C. C. Aggarwal & C. Zhai (Eds.), *Mining Text Data* (2012nd ed., pp. 77–128). Boston, MA: Springer US.

Aherne, F. J., Thacker, N. A., & Rockett, P. I. (1998). The Bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika, 34*(4), 363–368.

Aldenderfer, M. S., & Blashfield, R. K. (2006). *Cluster analysis* (22nd ed.). *Quantitative applications in the social sciences: Vol. 44*. Newbury Park, Calif: Sage Publ.

Anderson, T. W. (1963). Asymptotic Theory for Principal Component Analysis. *The Annals of Mathematical Statistics, 34*(1), 122–148.

Atkinson, A. B. (1970). On the measurement of inequality. *Journal of Economic Theory, 2*(3), 244–263. doi:10.1016/0022-0531(70)90039-6

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research, 3*(1), 993–1022.

Crawley, C. E. (2007). Localized Debates of Agricultural Biotechnology in Community Newspapers: A Quantitative Content Analysis of Media Frames and Sources. *Science Communication, 28*(3), 314–346.

Daily Hansard. (2010). *House of Commons: Debate on Thursday, 28 October 2010*. The Daily Hansard. Retrieved from http://www.publications.parliament.uk/pa/cm201011/cmhansrd/cm101028/debtext/101028-0001.htm

David, C. C., Atun, J. M. L., & La Viña, A. G. (2010). Framing the population debate: A comparison of source and news frames in the Philippines. *Asian Journal of Communication, 20*(3), 337–353. doi:10.1080/01292981003802168

Efron, B. (1982). *The jackknife, the bootstrap, and other resampling plans*. *CBMS-NSF Regional conference series in applied mathematics: Vol. 38*. Philadelphia, Pa: Society for Industrial and Applied Mathematics (SIAM 3600 Market Street Floor 6 Philadelphia PA 19104).

Entman, R. M. (1993). Framing: Toward Clarification of a Fractured Paradigm. *Journal of Communication, 43*(4), 51–58.

Fürnkranz, J. (1998). *A Study Using n-gram Features for Text Categorization.* Technical Report OEFAI-TR-98-30. Vienna, A.

Hauke, J., & Kossowski, T. (2011). Comparison of Values of Pearson's and Spearman's Correlation Coefficients on the Same Sets of Data. *Quaestiones Geographicae, 30*(2). doi:10.2478/v10117-011-0021-1

Hellman, H. (2001). Diversity - An End in Itself?: Developing a Multi-Measure Methodology of Television Programme Variety Studies. *European Journal of Communication, 16*(2), 181–208. doi:10.1177/0267323101016002003

Hofmann, T. (1999). Probabilistic Latent Semantic Analysis. In K. Laskey & H. Prade (Eds.): *UAI'99, Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (pp. 289–296). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hogenraad, R., Mckenzie, D. P., & Péladeau, N. (2003). Force and Influence in Content Analysis: The Production of New Social Knowledge. *Quality and Quantity, 37*(3), 221–238. doi:10.1023/A:1024401325472

Huson, L. W. (2007). Performance of Come Correlation Coefficients When Applied to Zero-Clustered Data. *Journal of Modern Applied Statistical Methods, 6*(2), Article 17. Retrieved from http://digitalcommons.wayne.edu/jmasm/vol6/iss2/17

Kohring, M., & Matthes, J. (2002). The face(t)s of biotech in the nineties: How the German press framed modern biotechnology. *Public Understanding of Science, 11*(2), 143–154. doi:10.1088/0963-6625/11/2/304

Kriesi, H., Grande, E., Lachat, R., Dolezal, M., Bornschier, S., & Frey, T. (2006). Globalization and the transformation of the national political space: Six European countries compared. *European Journal of Political Research, 45*(6), 921–956. doi:10.1111/j.1475-6765.2006.00644.x

Krippendorff, K. (2004). *Content analysis: An introduction to its methodology*. Thousand Oaks, Calif: Sage Publ.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review, 104*(2), 211–240. doi:10.1037/0033-295X.104.2.211

Laver, M., Benoit, K., & Garry, J. (2003). Extracting Policy Positions from Political Texts Using Words as Data. *American Political Science Review, 97*(02). doi:10.1017/S0003055403000698

Li, Y., Chung, S. M., & Holt, J. D. (2008). Text document clustering based on frequent word meaning sequences. *Data & Knowledge Engineering, 64*(1), 381–404. doi:10.1016/j.datak.2007.08.001

Lombard, M., Snyder-Duch, J., & Bracken, C. C. (2002). Content Analysis in Mass Communication: Assessment and Reporting of Intercoder Reliability. *Human Communication Research, 28*(4), 587–604.

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, Mass: MIT Press.

Marten, R. (2014). *Is it really about Facts?: The positive Side of 'Meforming' or turning Self-Disclosure to Social Capital in Enterprise Social Media*. Twenty Second European Conference on Information Systems, Tel Aviv.

Matthes, J. (2007). *Framing-Effekte: Zum Einfluss der Politikberichterstattung auf die Einstellungen der Rezipienten* (1st ed.). *Reihe Rezeptionsforschung: Vol. 13*. Baden-Baden: Nomos.

Matthes, J., & Kohring, M. (2008). The Content Analysis of Media Frames: Toward Improving Reliability and Validity. *Journal of Communication, 58*(2), 258–279. doi:10.1111/j.1460-2466.2008.00384.x

Miller, M., & Riechert, B. P. (2001). Frame Mapping: A Quantitative Method for Investigating Issues in the Public Sphere. In M. D. West (Ed.), *Progress in communication sciences: Vol. 16. Theory, method, and practice in computer content analysis* (pp. 61–75). Westport, Conn: Ablex Publ.

Miller, M., Andsager, J. L., & Riechert, B. P. (1998). Framing Candidates in Presidential Primaries: Issues and Images in Press Releases and News Coverage. *Journalism & Mass Communication Quarterly, 75*(2), 312–324.

Podani, J. (2000). *Introduction to the exploration of multivariate biological data*. Leiden: Backhuys.

Quinn, K. M., Monroe, B. L., Colaresi, M., Crespin, M. H., & Radev, D. R. (2010). How to Analyze Political Attention with Minimal Assumptions and Costs. *American Journal of Political Science, 54*(1), 209–228. doi:10.1111/j.1540-5907.2009.00427.x

Riloff, E. (1995). Little Words Can Make a Big Difference for Text Classification. In E. A. Fox, P. Ingwersen, & R. Fidel (Eds.): SIGIR '95, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 130–136). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/215206.215349

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), 513–523. doi:10.1016/0306-4573(88)90021-0

Scott, W. A. (1955). Reliability of Content Analysis: The Case of Nominal Scale Coding. *Public Opinion Quarterly, 19*(3), 323–325.

Semetko, H. A., & Valkenburg, P. M. (2000). Framing European politics: A content analysis of press and television news. *Journal of Communication, 50*(2), 93–109. doi:10.1111/j.1460-2466.2000.tb02843.x

Shih, L., Rennie, Jason D. M., Chang, Y.-H., & Karger, D. R. (2003). Text Bundling: Statistics-Based Data Reduction. In T. Fawcett & N. Mishra (Eds.), *Proceedings of the Twentieth International*

Conference on Machine Learning. [August 21 - 24, 2003, Washington, DC, USA] (pp. 696–703). Menlo Park, Calif: AAAI Press.

Simon, A. F. (2001). A Unified Method for Analyzing Media Framing. In R. P. Hart (Ed.), *Communication in U.S. elections. New agendas* (pp. 75–90). Lanham, Md: Rowman & Littlefield.

Sirkin, R. M. (2006). *Statistics for the social sciences* (3. ed). Thousand Oaks, Calif: Sage Publ.

Veltri, G. A., & Suerdem, A. K. (2013). Worldviews and discursive construction of GMO-related risk perceptions in Turkey. *Public Understanding of Science, 22*(2), 137–154. doi:10.1177/0963662511423334

Xu, W., Liu, X., & Gong, Y. (2003, July). *Document Clustering Based on Non-negative Matrix Factorization*. SIGR, Toronto, Canada.

Zegers, F. E., & Ten Berge, J. M. (1985). A Family of Association Coefficients for Metric Scales. *Psychometrika, 50*(1), 17–24.

Appendix

Normalize count matrix $C > C^n$
While $N > 1$:
    Compute distances between all pairs of vectors
    Combine pair with least distance d_{min} to form a cluster
    Compute the cluster gravity center
    Compute the radius of the new cluster r
    If $r > d_{min}$: tag this step as harmful
    Normalize cluster gravity center
    Add cluster as new vector to C^n
    Remove both combined vectors from C^n

For all steps tagged as harmful:
    Extract constituting cluster(s) which were not the result of a harmful step

Figure 1: Summary of the cluster analysis procedure in pseudo-code. The routine continues adding items and clusters to already existing clusters until the count matrix is reduced to a single cluster comprising all elements. Upon completion, the tree is not regarded as a solution but is further processed by selecting all homogeneous clusters from the solution that would increase in heterogeneity by the addition of the nearest element.

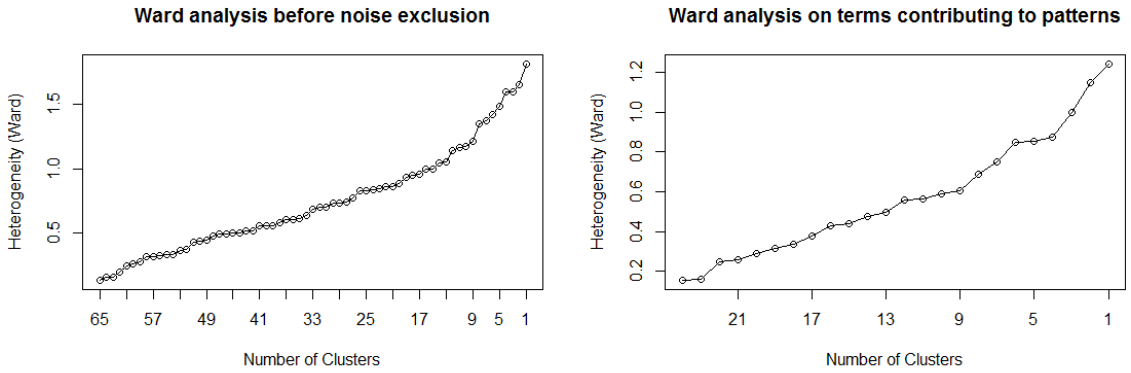


Figure 2: Elbow plots visualizing the increase of heterogeneity during Ward's analysis of the TDM representing the first phase. Left: Ward's analysis of the normalized TDM. There are weak elbows indicating 3, 9, 13, 21, or 26 clusters. Right: Ward's analysis on the terms remaining after HECANE eliminated random noise. A strong elbow indicates four clusters, while weaker ones again indicate 9 or 13 clusters.

Table 1: Clusters in the first phase of the debate.
In this table, frame constituents making up each cluster in the first phase of the debate (before Oct. 15) are listed. While some clusters include all kinds of frame constituents, some are missing problem definitions, causes, effects, or treatments. In all cases, at least two different types are present.

Frame	Content
1-1	Problem definitions: Unemployment in general, unemployment rate and job situation abroad. Cause: Business closures Effect: Party Politics
1-2	Cause: Welfare policy Treatment: Decrease financial support, increase education, Improve job service, redistribute budget
1-3	Problem definitions: Domestic unemployment, sinking structural unemployment Cause: Structure of the labor market Effect: Reduction of consequences for the domestic economy Treatments: Increase financial support of unemployed persons, general reform of the benefit system
1-4	Problem definitions: Domestic unemployment rate, job loss of individuals Causes: Mass redundancies and rehiring in private sector, other causes Effects: Psychological stress and anxiety of individuals Treatment: More restrictive unemployment benefits

Table 2: Clusters in the second phase of the debate.
In this table, frame constituents making up each cluster after the announcement of public sector job cuts (after Oct. 29) are listed. While some clusters include all kinds of frame constituents, some are missing problem definitions, causes, effects, or treatments. In all cases, at least two different types are present. PD: Problem definitions; C: Causes; E: Effects; T: Treatments.

Frame	Content
2-1	PD: Unemployment in General, changing rates of unemployment abroad, Job situation abroad, domestic employment rate.
2-2	PD: Unemployment and job loss of single persons, reduction of jobs, pessimistic outlook C: Budget cuts, Restructuring of the labor market, business closures, other causes E: Other effects
2-3	PD: Unemployment in general, shortage of jobs abroad C: Economic crisis E: Party politics
2-4	PD: Youth unemployment C: Financial situation T: Reduction of financial support, the unemployed must make an effort
2-5	C: Structure of the labor market T: Increase financial support, make re-entry to the labor market more attractive, business regulations, improve education, reforms of the unemployment benefit system
2-6	C: Outsourcing, migration, life quality of the unemployed E: Consequences for the domestic economy and for communities T: Redistribution of budget
2-7	C: Hiring policies in business Effects: Strikes, Emigration of the workforce, negative consequences for business

Table 3: Overview of cluster solutions obtained in the three analyses performed.
For each analysis, the total number of terms entered into the analysis, the clusters resulting from the analysis, and the share of stop words and noise are reported. In Analysis 1, only terms appearing in >1% of the tweets are included. In Analysis 2, this threshold was lowered to 0.5%, and in Analysis 3 it is 0.25%.
For the second and third analysis, information on the agreement with the previous analysis or analyses, respectively, is provided.

	Analysis 1		Analysis 2		Analysis 3	
	Terms	Clusters	Terms	Clusters	Terms	Clusters
Total count	229	25	516	58	1080	122
Stop words	72	11	110	17	124	19
Meaningful clusters	105	14	310	41	718	103
Noise	52	0	96	0	238	0
Re-allocation in sub-clusters from analysis 1	-	-	134	47	148	78
Re-allocation in sub-clusters from analysis 1	-	-	-	-	321	105

Table 4: 14 Homogeneous and meaningful clusters of terms identified in 15,950 tweets of US politicians in March/April 2014.
This table only represents the result for the first analysis, using terms that were present in more than 1% of all tweets (N = 229). For each cluster, a comprehensive name and the full content are provided.

Cluster	Content
Functional Tweets	
Events	event, today, out, next, yesterday
Relativization	some, my, don't, that, but
Beginning	first, week, happy, big, way, good
Attack	stand, fight, senator, sen
Sympathy	thank, follow, know, best, love, too
Contact	office, state, service, community, please, check, open, call, number
Interview	and I, CNN, book, debate, obama, sign, talk, be on, i'll, to talk, talk about, listen, forward, look forward to
Issue Tweets	
Home	house, better, question, over, visit
Veteran Day	veteran, off, home, day, work, all, proud
Senate Bill	senate, sure, bill, against, reform, law, need, end
Family	family, dc, vote, stop
Budget	america, gop, congress, must, budget
Student Health Care	student, health, care, keep, support, job, school, fund, protect, to help
Obamacare Debate	obamacare, I will, tune, discuss, to discuss

Anhang B1: Angrist

Online-Ressource:

Wettstein, M. (2014). *Angrist 1.2: Documentation and Reference for the Coder Interface*. Online verfügbar: http://www.ipmz.uzh.ch/Abteilungen/Medienpsychologie/Recource/Angrist/ANGRIST_1-2-en.pdf

Im Folgenden wird die Version bei Einreichung dieser Dissertation dargestellt. Eine vollständige und ständig aktualisierte Version der Dokumentation und Anleitung, inklusive Beispielprogramme, sind Online verfügbar unter:

http://www.ipmz.uzh.ch/de/Abteilungen/Medienpsychologie/Recource

angrist 1.2(en)

Documentation and reference for the coder interface

Martin Wettstein

angrist, n ['æŋgrist] 1. Python script to generate a query form for relational data input in content analyses. It is especially useful for the application of hierarchical codebooks, as it allows data entry on different levels of analysis without raising the cognitive load of coders. 2. (from Sindarin: '*Iron cleaver*'): Legendary knive from the ballad 'The Lay of Leithian' by J.R.R. Tolkien. The young hero Beren uses this knive to descend to the lowest levels of hell in his quest to separate a precious gem from Morgoth's iron crown.

Zürich, May 2014

Current version available at:

http://www.tarlanc.ch/angrist/ANGRIST_Dokumentation.pdf

Inhalt

1. Introduction ... 3
1.1. Area of application ... 3
1.2. Information for Users ... 4
1.3. Technical detail ... 4
2. Angrist GUI ... 5
2.1. Question-Types ... 7
2.1.1. Help Text ... 9
2.1.2. Example ... 9
2.2. Behind the Scenes ... 11
3. Required files ... 13
3.1. angrist.py ... 13
3.2. a_codebook.ini ... 13
3.3. a_settings.ini ... 14
3.4. Todo-List ... 14
3.5. Text folder ... 15
3.6. Python Compiler ... 15
4. Important variables ... 16
4.1. settings ... 16
4.2. Storage ... 17
4.3. Codebook ... 19
4.4. def_val ... 19
4.5. prog_pos ... 19
4.6. dta_pos ... 19
5. Setting up Angrist ... 21
5.1. Making a plan ... 21
5.2. Definition of the codebook ... 21
5.3. fuellen() ... 23
5.4. ask() ... 24
5.4.1. Data storage ... 24
5.4.2. Calling functions using button questions ... 25
5.5. submit() ... 25
5.5.1. Storing values ... 25
5.5.2. Changing the level of analysis ... 26
5.5.3. Cleaning up the page ... 27
5.5.4. Complete submit function ... 27
5.6. abort() ... 28
5.7. back() ... 28
5.8. rb_tamper() ... 29
6. Extension: Aeglos ... 30
7. Glossary ... 31
7.1. Initialization ... 31
self.fuellen() ... 31
self.set_window() ... 31
7.2. Project specific functions ... 31
self.ask() ... 31
self.submit(overspill=0) ... 31
self.abort() ... 32
self.back() ... 32
self.rb_tamper() ... 32
7.3. Important aides ... 32
self.show_review(level,rm=1,edit=0,height=3) ... 32
self.hide_review() ... 33
self.remove_item(level, height) ... 33
self.edit_item(level) ... 33
self.level_up() ... 33
self.level_down(variable,level) ... 33

self.buttons(check=1,abort=0,back=1,pause=1) 34
self.check_entries() 34
self.clean_up(pos, typ='') 34
self.clean_up_all() 35
self.codegetter(variabel, item) 35
self.namegetter(variable, item) 35
self.unit_select(level): 35
self.unit_confirm(level,sel_tags=[]): 35
self.store_var_all(setdef=0) 36
self.store_var(variable,pos=-1,setdef=0,store=1) 36
self.message(m_id,m_type=1,variable='Err_Msg') 36
7.4. Question functions 37
self.question_dd(var,pos,width=40) 37
self.question_txt(var,pos,width=40) 38
self.question_txt2(var,pos,width=40,height=3,getselect=0) 38
self.question_cb(var,pos,layout='hor',defval=0) 39
self.question_rb(var,pos,layout='vert',defval='98') 39
self.question_sd(var,pos,points=5,defval=0) 40
self.question_rating(var, pos, scalelist=['disagree','','','','agree'], valuelist=['1','2','3','4','5'], defval='1') 40
self.question_bt(var,pos) 41
self.question_ls(var,liste,multi=1) 41
self.question_lseek(var,liste,multi=0) 42
self.question_ladd(var,liste,multi=0) 42
self.question_mark_units(var,level) 42
self.question_sel_units(var,level) 43
self.question_menu(var,position) 43
7.5. Basic functions 44
baum_schreiben(direc) 44
bereinigen(uml_string,lc=0,lb=0,uml=0) 44
self.clean_all_tags(sel_tag=[]): 44
self.ausblenden() 44
7.6. Input and output of data 44
self.export_data(dta_pos_all, varlist, filename, debug=0) 45
baum_export() 45
artikelholen(ID) 45
textmine(linelist) 45
get_codebook(filename) 45
get_todo(filename) 45
check_todo(filename) 45
get_data(filename, varlist=[]) 46
get_varnames(filename) 46
write_data(data, varlist, filename) 46
8. Impressum 47

[..]and tried its hard edge, bitter-cold,
o'er which in Nogrod songs had rolled
of dwarvish armourers singing slow
to hammer-music long ago.
Iron as tender wood it clove
and mail as woof of loom it rove.
The claws of iron that held the gem,
it bit them through and sundered them;[..]

(J.R.R. Tolkien -The Lay of Leithian, 4143-4151)

1. Introduction

Angrist[1] is a highly adaptable script for displaying query inputs for quantitative content analysis. The special feature of this script is the ability to handle hierarchically structured codebooks and allows for the simultaneous coding on different levels of analysis within a given text. Accordingly, the data is exported in a relational database, pseudo-XML, or JSON-format, depending on the preferences and needs of the content analysis. By organizing the relational data input in linear and easy-to-accomplish tasks, data input is feasible even for highly complex codebooks without risking cognitive overload in coders.

Neither the data structure nor the complexity of the codebook has to be considered by the coder. Instead, the coder is confronted with a series of questions in natural language which may be answered from sets of previously defined items. The codes, storage, and interconnectedness of the variables are hidden from the coder to minimize cognitive load. For the task of answering the questions, Angrist provides a wide range of question-types, including lists, dropdown-menues, checkboxes, and rating-scales.

Since the coder interface runs as a stand-alone Python-script filtering of questions and pages is very easy and the tool is expandable with a wide variety of modules for data in- and output, as well as data processing, available for Python. One module specificially designed for the combination with Angrist even allows for the automated coding of certain variables by means of a learning algorithm which is automatically trained by manual data input.

1.1. Area of application

Angrist is especially useful in the analysis of texts using hierarchical codebooks. A wide variety of content analyses in different fields have made use of the coder interface so far, three prototypical ones of which are referred here:

- In the course of a project aiming for the quantitative analysis of public debates, articles have been divided to the statements of different actors, including the journalist himself. Each statement was again divided to single problem definitions, causes, evaluations, and treatments it contained. Accordingly, each text was analyzed on three levels of analysis (Text, Speaker, Statements). In total, more than 9000 texts have been analyzed, the largest of which containing more than 40 units of analysis.
- In the course of a project analyzing financial news, all financial developments which are part of the index SNP500 have been identified in articles within the finance-section of influential newspapers. For each development, all causes, consequences, and evaluations by actors have been coded.

[1] **A**djustable **N**on-commercial **G**adget for **R**elational data **I**nput in **S**equential **T**asks

- For the analysis of online comments, the comment threads to online news coverage have been investigated. Each comment was thereby divided to single references to other discussants, the topic of the article, or persons outside the discussion. A total of 250 articles with more than 11'000 comments and 16'000 references have been coded in this analysis. The coding of a single comment with all references thereby took less than three minutes.

All of these examples used three different levels of analysis. The script is expandable and would allow more levels of analysis. The impact of a higher complexity on the coding process and cognitive load of the coder has to be kept in mind, however. Also, as no analysis has ever used more than four levels of analysis in Angrist, the sound function of all features has to be tested before applying a codebook with more than four levels.

1.2. Information for Users

Angrist was designed as an Open Source academic Software and is subject to according regulations (http://www.opensource.org/docs/osd). All parts of this script may be copied, redistributed, changed, and adapted to personal tastes or needs.

In spite of intense care and rigorous testing of Angrist, programming errors may not be ruled out. This means that even correct use without changes to the program code may lead to damage to entered data or suspension of the function of this program; -although this is highly improbable. In the case of mishandling of this tool or changes in the program code, the probability of data loss increases.

The programmer declines any responsibility of the loss, damage or incorrect acquisition of data and is not liable to amend programming errors. In order to prevent damage to the program and data, the modification of the script should be limited to the part specific for the project. Modifications of built-in functions should only be done by well-trained persons.

The programmer is doing his best to correct any programming errors and to update the script on a regular base. The current version of the program may be found at:

http://www.ipmz.uzh.ch/Abteilungen/Medienpsychologie/Recource/Angrist.html

For questions and support, please do not hesitate to contact the programmer directly.

1.3. Technical detail

Angrist was written in python 2.7 (from version 1.1, the script also runs in Python 3.x) (www.python.org) and uses the TK-interface (library: tkinter) for displaying the GUI. The GUI is designed for Windows and does not run properly on Unix and Mac. For usage on these OS, please use a windows emulator.

The tool may display the text to be coded on screen. This display is currently limited to ASCII-Textfiles, the import of images, PDF-files and rich-text is planned for future versions, however. Please convert texts to be coded to ASCII-format prior to using them as text sources. Unicode encrypted documents may not be displayed properly.

2. Angrist GUI

The graphical user interface (GUI) of angrist is a single dialog window presenting the coder with a series of questions. The questions are arranged in subsequently presented pages which may be set up dynamically. On each page a maximum of three different questions may be asked, using different answering schemes such as dropdown-lists, lists, checkboxes, rating scales, or buttons.

Next to the question page, the coder may read the text to be coded and highlight portions of the text using predefined colors. All highlighted portions of the text may be read by the program and used as input for the content analysis.

The layout of the GUI is defined using the function `self.set_window()` and remains fixed for the whole analysis. Modifications of the layout may be done using this function and will affect all pages of the GUI. The standard design (see Figure 1) will display the pages on the left and the text to be coded on the right side of the dialog. The check-button, which is used to submit answers, is located at the bottom-left of each page. Since this design has been found to be irritating to some right-handed coders, the layout may be flipped using the setting `'Layout'`.

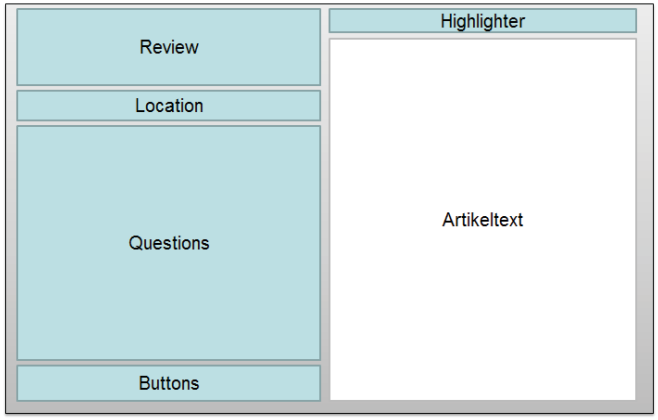


Figure 1: Standard-Layout of the Angrist GUI

Artikeltext

In this text area, the current text is displayed. The source for this text is an ASCII-formatted Text-file which is named after the ID of the current text.

Portions of this text may be highlighted for the purpose of navigation and data input. In most analyses, this feature is used to mark the units of analysis which should be coded in this text. Angrist may use the highlighted portions as input to guide the coder through the coding process.

If no text is available, this part of the dialog will not be used and the dialog only needs one half of the screen. The other half may be used for displaying texts or audiovisual documents using other applications.

Highlighter

When text is available, a menu of markers is displayed above. These markers may be used to highlight any portion of text with a given color. The colors, labels and the meaning of these buttons may be

defined using the setting `'Highlight_Buttons'`. In analyses using multiple levels of analysis, the markers may be used to highlight units of analysis within the text for later coding.

Review

At the top of each page, a frame is reserved for a review of previous data input. When visible, a list displays all units of analysis within a given level below the current level of analysis. This review-list may be called using the function `self.show_review()` and will be shown only on pages explicitly demanding it. On all other pages the list is disabled.

The elements within the review-list may be removed or edited by the coder if allowed. While removal is trivial to the program, editing of single items within the list requires some preparation. The function `self.edit_item()` specifies the pages which should be called when editing an item on a given level of analysis. The names of these pages have to be specified for each project.

Location

In the frame beneath the review-list, a text area is displayed. This area may be used to inform the coder on his current location in the coding process. The function `self.locate()` may be used to display information on each level of analysis.

Questions

The questions-frame is the central frame on each page. Within this frame, three questions and answering devices may be displayed to the coder. The questions may be called by using the `self.question_xx()`-functions. Each of these functions will display the question for a given variable at the specified position. Upon submitting the answers, this frame is cleaned and may be filled with new questions on the next page.

Please consider that there are question-types which need more space than others. Especially checkbox-lists or long series of rating-scales are consuming lot of the space available. While the GUI automatically reshapes to take in the whole page, this may lead to a window-size exceeding the size of notebook monitors. When using more than one Checkbox, Radiobutton or Rating-Question on one page, you might want to check the size of the page carefully.

Buttons

The bottom frame on each page contains up to four buttons with fixed purpose.

Check-Button: This button is linked to the function `self.submit()`. The submit-function is used to store current entries and set the program position for the next page. This button is vitally important on all pages except for pages which have other means to call the submit-function (e.g. a Buttons-question).

Abort-Button: This button is linked to the function `self.abort()`. The abort-function may be defined individually for each page where this button is available. Usually it is used to cancel the coding of a unit of analysis and return to the decision point.

Back-Button: This button is linked to the function `self.back()`. This function will set the program position of the last page visited and direct the coder there without storing current entries. Be aware that this function just goes back one page. For pages where the back-button should lead to entering another level of analysis, this has to be specified.

Break-Button: This button is linked to the function `self.pause()`. This function blanks out the page and start the timer for break-time. At the end of the coding, the break-time is stored to the data to keep track of net duration of the coders' work.

2.1. Question-Types

The GUI is able to present the coder with a wide variety of question types. These questions are called within the ASK-Function for all pages of the questionnaire. Each of the different question-types may be called using the appropriate `self.question_xx()`-Function. These functions take the codebook-variable and the position of the question as arguments and place the demanded question-type on the page. In this chapter, a short summary for all question-types is provided.

Dropdown

Dropdown questions present the coder with a list of options as defined in the codebook from which he may select the appropriate one. While each option has a code and a label in the codebook, only the label will be presented to the coder as an option. When calling a dropdown question, the first option in the list will be set as default value if no other default value has been specified.

When submitting the answer to a dropdown question, the code of the option will be stored to the data as a string variable. If this code happens to be '98', the answer is not accepted and the coder is prompted to select another option. Therefore, you may use the option 'Please select an option...' with code '98' as first option in the list. The coder will then be forced to make a choice to continue.

Dropdown questions are called using the function `self.question_dd(var,pos)` where `var` is the name of the variable as stored in the codebook and `pos` is an integer ranging from 1 to 3 indicating the position within the questions frame.

Text

Text questions ask for a manual input by the coder. This input may be entered to a single-lined textfile or to a large textbox. The values are stored as strings. Even if there are options for this variable in the codebook, these options will be ignored when calling a text question.

Text questions may be called using the function `self.question_txt(var,pos[,width])` for single-line text entries or `self.question_txt2(var,pos[,width[,height[,getsel]]])` for large textboxes. `var` is the name of the variable as stored in the codebook and `pos` is an integer ranging from 1 to 3 indicating the position within the questions frame. The width and height of textboxes may be changed using the corresponding arguments. The argument getsel is an integer with value 1 or 0 which indicates whether it should be possible to copy/paste a part of the displayed text by means of a button.

Checkbox

Checkbos questions may be answered by checking the options provided in the codebook. Up to 14 options may be defined in the codebook and will be displayed next to checkboxes.

When storing the answer to a checkbox-question, a dummy variable for each option is created which may take the values 0 (not selected) and 1 (selected).

Checkbox questions are called using the function `self.question_cb(var,pos)` where `var` is the name of the variable as stored in the codebook and `pos` is an integer ranging from 1 to 3 indicating the position within the questions frame.

Radiobuttons

Radiobutton questions may be used analogous to dropdown questions. Here, however, all options will be visible at the same time and the coder may select any one of the options as an answer.

When changing the value of a radiobutton question, the function self.rb_tamper() is called automatically. You may use this function to define an action which is to be taken upon selecting certain

values. For example, you may choose to ask for the name of the news agency if the coder selects the value 'News Agency' from a list of options.

Radiobutton questions are called using the function `self.question_rb(var,pos)` where `var` is the name of the variable as stored in the codebook and `pos` is an integer ranging from 1 to 3 indicating the position within the questions frame.

Buttons

Buttons question present the coder with up to four different choices which he may select by pressing the corresponding button. Each button may be linked to a function (either existing or custom) which will be called as an answer to the coders' decision.

No value is stored for buttons questions. They are usually employed at decision-points where the choice of a coder may lead to different courses in the coding process. Each event has to be defined when calling a buttons question.

Buttons questions are called using the function `self.question_bt(var,pos)` where `var` is the name of the variable as stored in the codebook and `pos` is an integer ranging from 1 to 3 indicating the position within the questions frame.

Rating

Rating questions present the coder with a list of items to be rated on a scale. By default, a 5-point likert scale, ranging from 'disagree' to 'agree' is used. The number of points, as well as their values and labels may be defined when calling a rating question.

When submitting the answers to a rating question, dummy variables for each option will be created which may take any of the values specified for the rating scale.

Rating questions are called using the function `self.question_rating(var,pos,scalelist, valuelist,defval)`, where `var` and `pos` again indicate variable and position. The parameter `scalelist` is a list containing all labels for the points of the rating scale. The parameter `valuelist` is a list containing the corresponding values. These lists have to have the same length. The number of points of the rating scale is equal to the length of these lists. If desired, a default value may be set using the parameter `defval`. If this value is off the scale (i.e.: not in the valuelist), no selection will be made on default.

Semantic Differential

As with rating scales, semantic differential (SD) questions prompt the coder to rate an item on a scale. The scale, however, spans for each item from one option to another one. This question type may be used to ask for a rating of a text from 'boring' to 'interesting' and from 'emotional' to 'sober'. The code of each option (as defined in the codebook) will be used as first option, the label will serve as second option for each item.

When submitting the answer to a semantic differential, a new variable is created for each item with the value assigned on the rating scale.

SD questions are called using the function `self.question_sd(var,pos,points,defval)`, where `var` and `pos` again indicate variable and position. The parameter `points` is an integer indicating the number of points which is to be used. If desired, a default value may be set using the parameter `defval`. If this value is off the scale (i.e. larger than points) no selection will be made on default.

List

List questions may be used to present the coder with long lists of options from which either one or several items may be selected by the coder. If desired, the list may be searchable by entering charcters to a textbox above the list. This option is recommended for lists with more than 20 items. In addition, the coder may be prompted to add list items to a smaller list instead of simultaneously selecting them. This is recommended for tasks where several items from a list have to be chosen.

When submitting the answer to a list question, a list-variable is created from the selection. If not processed any further, a string of this list-variable will be stored in the data.

List questions may be called using the functions `self.question_ls(var,list,multi)` for simple lists, `self.question_lseek(..)` for searchable lists and `self.question_ladd(..)` for searchable lists which ask for the export of items to a smaller list. Other than all other question types the list question may only be placed at position 1. The parameter `list` indicates the variable which contains the list options. This variable may be different from the variable containing the question. The parameter `multi` may take the values 0 and 1 and indicates whether multiple selections are allowed.

2.1.1. Help Text

For each variable, a short help text may be defined in the codebook. When calling a question, a small question mark is displayed at the right border of the page. When clicking on this question mark, a small Popup window appears, containing the help text for this variable.

2.1.2. Example

As an example, we define the variable 'Author' in the codebook and call it using all available question types. The definition is as follows:

```
[Author]
Who is the author of this text?
Please indicate the author
The author of a text refers to the person responsible for the content and
    presentation. In most cases the author is a journalist of the source in which
    the text has been published or a journalist of a news agency.[...]
1:Journalist of the Medium
2:News Agency
3:Reader / Audience
4:National political actor (Chose from Appendix B)
5:Other
6:Not identifiable
```

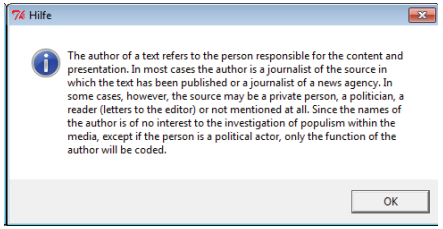
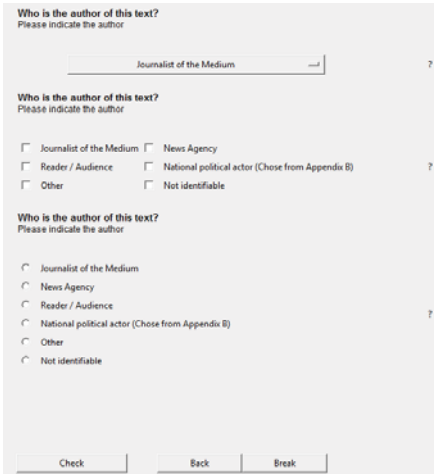
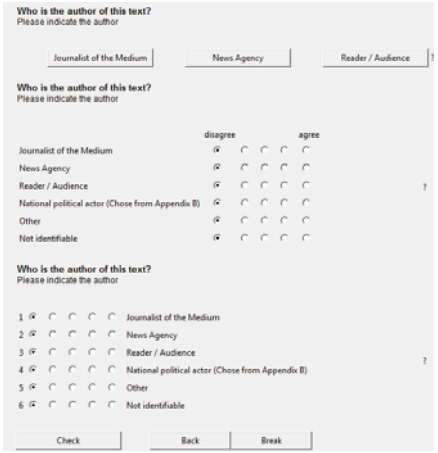


Figure 2: Help text which is displayed in a pop-up as soon as the coder clicks the question mark at the right-handed border of the page.

Page definition	Presentation

`self.buttons()`
`self.question_dd('Autor',1)`
`self.question_cb('Autor',2)`
`self.question_txt('Autor',3)`
	

`self.buttons()`
`self.question_bt('Author',1)`
`self.question_rating('Author',2)`
`self.question_sd('Author',3)`
	

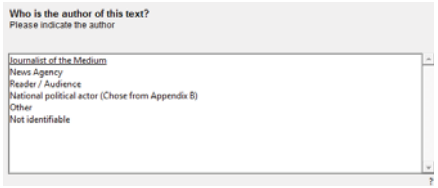
`self.buttons()` `self.question_ls('Author','Author')`	

Figure 3: Presentation of different question types for the same variable. As the variable was not designed for semantic differential questions, the example looks awkward for this question type. The coder has to rate the items on a scale ranging from the code of the item to its name.

2.2. Behind the Scenes

The script basically runs on one external file, two functions, and two global variables. All five constituents have to be adjusted to each other in order for the program to run smoothly and store all data correctly. In this chapter, a brief introduction to the constituents is provided. Please refer to the according chapters to find out more about each one.

The codebook (see chapter 3.2) is a file containing the questions to be asked and the answering options. Entries in the codebook may be changed independently from the rest of the program as long as the names of all variables remain unchanged. The ASK-Function contains a large IF...ELIF..ELSE block defining the questions to be asked on each page. All pages have an individual name and may contain the query for three variables within the codebook. The SUBMIT-Function contains a large IF...ELIF...ELSE-block which defines the actions to be taken upon submission of each page. Usually, the actions are storing, cleaning up and setting a new page. For each page, both an ASK- and a SUBMIT-handler have to be entered. The variable prog_pos contains the name of the current page. When running the ASK- or SUBMIT-function, only actions defined for the current value of prog_pos will be executed. Thus, the variable determines the exact position within the questionnaire. Finally, the variable dta_pos contains the current position within the data. This includes the names of the current level and the current unit of analysis as well as the levels and names of all units above the current unit of analysis. This variable determines where the program stores current entries. Figure 4: Illustration of the interconnectedness of all functions and variables in a short example. At two points, the entries made by the coder are influencing the coding process. First, by answering to Var1, Option A will lead to page p2 being displayed, Option B will lead the program to skip this page. Second, when answering F to Var3, an additional question (Var7) will be displayed on page p3.Figure 4 illustrates the coding process in a simple example.

When setting up Angrist, the codebook has to contain all variables used in the ASK-Function. Likewise, all pages have to be defined for the ASK and SUBMIT-function. If there are incongruences in the spelling of the names of pages and variables, the program will fail.

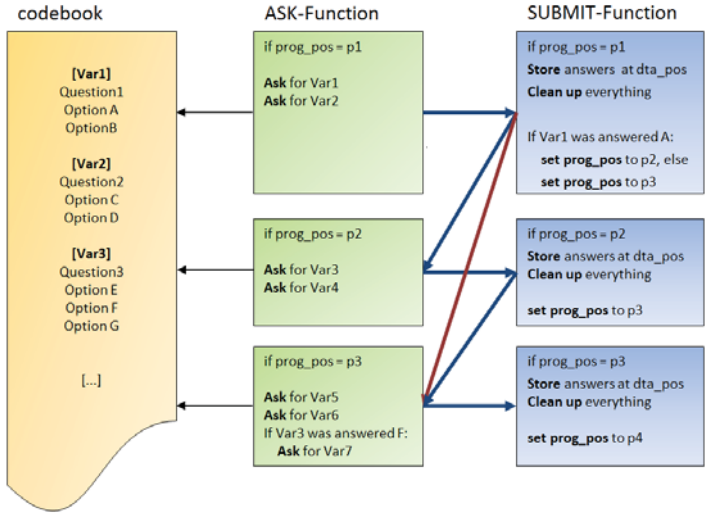


Figure 4: Illustration of the interconnectedness of all functions and variables in a short example. At two points, the entries made by the coder are influencing the coding process. First, by answering to Var1, Option A will lead to page p2 being displayed, Option B will lead the program to skip this page. Second, when answering F to Var3, an additional question (Var7) will be displayed on page p3.

3. Required files

Angrist was designed as a stand-alone python script. The program does only need built-in modules available in the standard configuration of Python 2.7. Accordingly, no additional scripts are needed for the execution of Angrist.

The minimal configuration for the script to work is the script (usually: `angrist.py`) itself and a codebook (usually: `a_codebook.ini`), which contains the questions and options for each variable. When both files are present in a directory and set up correctly, Angrist may be executed.

There are, however, two additional files which may be used to configure the tool, provide the script with additional information. An ini-file containing individual settings for each coder (usually: `a_settings.ini`) may be used to set the name of the coder, layout, presentation, and other parameters which vary between coders. A todo-list (usually `to_do.txt`) may be used to provide the coder with a list of text identifiers from which angrist may select the next text to be coded. Errors in the spelling of article identifiers may be prevented in this way.

In addition, a directory containing texts (usually: `Texts\`) may be used to store the texts in ACSII-format in order to be displayed during the analysis.

3.1. angrist.py

The file `angrist.py` is the central python script which initializes and controls the GUI for data input and stores the entered data in a specified format. The script has to be set up individually for each project and should run smoothly before starting the content analysis. During the phase of data acquisition, the script should not be modified any more.

The script was written in Python 2.7 but it also runs on Python 3. If no Python compiler is installed on a computer, an offline-version of python may be used from any portable device. The offline-version should contain the libraries *tkinter*, *os*, and *time*.

The modification of the script demands no advanced knowledge of Python. Basic operations, definition and calling of functions, as well as the application of methods to objects should be known, however. For a detailed reference of the programming language, see http://docs.python.org/tutorial/. For an introduction to the tkinter-library which is used to build the dialof for the GUI, see: http://www.pythonware.com/library/tkinter/introduction/

3.2. a_codebook.ini

The file `a_codebook.ini` is the main source of information for the script. The file contains standardized entries for each variable used in the analysis. Each entry contains a question to be asked, additional information for the coder, a help text, and a list of options to choose from.

The format of this file, which should be held in ASCII, is very simple and does not require any programming skills. For the correct execution of Angrist, the format of the codebook has to be followed strictly. The format is as follows:

```
[Name of the variable]
Question (1 line only)
Coder Information (1 line only)
Help Text (1 line only)
Code1:Item1
Code2:Item2
Code3:Item3
...
```

Each entry begins with the name of the variable in brackets [..] and ends with at least one empty line. Below the name of the variable there are three lines for text input. The first line contains the question to be asked. The second line contains coder information to be displayed beneath the question. The third line contains the text to be displayed if the coder requires a help text. Since each of these entries is limited to one line, no line-break is allowed in any of them. If you want to display line-breaks, use the number sign (`#`) to mark their location. When displaying text, these characters are equivalent to line-breaks in Angrist.

The list of options may contain any number of lines, each of which is used as an option. Each line is to be divided by a colon (`:`). The portion of the line left to the first colon is used as code of the option and will be stored to the data. The portion right to the first colon is the label of the option and will be displayed within the GUI. Both code and label may be any string of characters. If you omit the colon, label and code will both take the value of the whole item.

3.3. a_settings.ini

The file `a_settings.ini` contains individual information for each coder. This information usually includes the name of the coder, the preferred font and size, and the preferred layout. It may also contain a variable called `'Default-Values'` which contains defaults for some of the variables in the content analysis.

All information within this file will be included in the `settings` directory in Angrist. All settings which are defined in Angrist itself are overwritten if there is conflicting information in this settings-file. Thus this file may be used to change the set-up of Angrist for each coder individually. For example, it may be useful for the project leader to set the `'Debugging'` setting to 1 and for coders to set the `'Insecure'` setting to 1.

```
##Coder Informations

[Coder-Settings]
Individual Settings
All Options are stored in the settings-dictionary
All entries have the form Name:Value
Codierer:mw
Font:Arial
Layout:Righty
Fontsize:11
Insecure:0

[Default-Values]
Values to be retained
The default values are loaded at startup
All entries have the form Name of the Variable:Value
Source:3
```

3.4. Todo-List

The Todo-list is a list of filenames (with extension .txt) or article identifiers (without extension) on seperate lines which have to be coded. Angrist will take the first item from this list upon initialization and load the corresponding text from the text folder. Upon completion of a text, the entry is removed from the list.

Hint: The easy way to create a todo-list from a folder full of text-files: Create a textfile which contains the single line: `*.txt /b >..\to_do.txt` in the text folder. Change the extension of this file to `.BAT` and execute it by double-clicking it in windows. The command creates a todo-list in the parent directory containing all filenames on seperate lines.

3.5. Text folder

The text folder is a sub-directory (usually in the same directory as angrist.py) which contains all texts to be coded. The texts should be stored as ASCII Textfiles with the extension .txt. The name of the text folder may be changed using the settings-file.

3.6. Python Compiler

Since Angrist is not an executable but a python script, a python compiler is needed in order to run it. Python 2.7 is recommended but Angrist also runs on Python 3. If python may not be installed on the computer, an offline version (i.e. the Python directory from a computer where it was installed) may be used from any directory on the computer or from a portable device.

You may use the command `PATH\Python27\python.exe angrist.py` to execute Angrist with an offline version of the compiler. `PATH` is the path of the python directory.

4. Important variables

The script uses a large number of variables for internal storage and data management. For the programming and handling of the script, only few of them have to be known and understood. The most central variables are the `settings`-dictionary, the `storage`-dictionary and the position variables (`prog_pos` and `dta_pos`).

4.1. settings

`settings` is a global variable of the type dictionary[2] which contains all global settings of the script. There is a number of fixed keys within this dictionary which may be used to change the appearance and handling of the GUI and the program in the background. Next to these central settings, each project may define any number of additional settings as needed or desired.

The list below shows the most central settings which are required for built-in functions and should therefore not be overwritten by project specific variables:

Name	Variable Type	Purpose
Coder	String	The name of the coder. Used in any data output. Default: `'default'`
Font	String	The font used in text areas. Default: `'Arial'`
Fontsize	String	The size of the font of displayed texts. Default: `12`
Layout	String	Controls Right- or Left-handed layout. Default: `'Lefty'`
Curr_Page	List of Lists	Contains three lists with two items each that control the layout of each page. Do not manipulate!
Page_History	List of Strings	Contains a list of all pages visited in this coding session. Do not manipulate!
Input	List of SysVars	Contains dynamic variables with information on current input. Do not manipulate!
Path_Log	String	Contains the log-file of functions called in this coding session. Used for debugging.
Verb_Log	String	Contains the log-file of all text output. Used for debugging.
Verbose	String ('0'/'1'/'2')	Controls the verbosity of the program. If set to `0`, no text output is generated. If set to `1` only important events are written in the logfile. If set to `2`, the program is maximally verbose and comments all steps. Value `2` is used for debugging only.
Debugging	String ('0'/'1')	If set to `0` the GUI is set for coding. If set to `1`, the GUI may be used for debugging. In this mode all entries will be accepted and you may easily access `settings['Path_Log']` and `settings['Verb_Log']`
AEGLOS	String ('0'/'1')	If set to `1`, the aeglos module is used for automated content analysis in the background to set default values. Only use if aeglos has been installed and trained.
Insecure	String ('0'/'1')	If set to `1`, the coder may report insecurity regarding a decision. Only use in coder training.
Multi_Items	List of strings	Contains the list of variables which use dummy-variables in storage (checkbox-, sd-, and rating-questions). These variables will be

[2] In Python sind dictionaries eine Klasse von Variablen, in welchen Werte für unterschiedliche Variablen abgelegt sind. Sie können auch als indizierte Listen verstanden werden. Siehe http://docs.python.org/library/stdtypes.html#dict für weitere Informationen.

		provided extra space in the data.
Break	Integer	Contains the timestamp of the beginning of the current break. When not in break, the value is set to `0`.
Break_Time	Integer	Contains the total duration of breaks in this session.
Coding_Time	Integer	Contains the total duration of coding in this session.
Hotwords	Dictionary	Contains a list of keywords which may be highlighted in the text.
Auto_Highlight	String ('0'/'1')	Controls whether the Hotwords should be highlighted upon startup.
Todo	String	Defines the name of the todo-file. Default: `'to_do.txt'`. If the value is set to an empty string, no todo list is used.
Package_Todo	String	Defines the name of the todo-file containing the names of folders each containing another todo-list and a set of texts. Useful for assigning packages of articles to the coders. If set to an empty string no package todo list is used. Default: `''`
Codebook	String	Defines the name of the codebook file. Default: `'a_codebook.ini'`. If set to an empty string, the file `'a_codebook.ini'` will be used as codebook file.
Settings	String	Defines the name of the settings file. Default: `'a_settings.ini'`. If set to an empty string, the file `'a_settings.ini'` will be used as settings file.
Text_Folder	String	Defines the folder which contains the texts. Default: `'Texts\\'` (mark that backslashes must be escaped!)
Out_Track	String	Defines the name of the tracking-file. All steps in the coding process are logged to this file. Useful for debugging. If set to an empty string, no tracking is used. Default: `''`
Out_Tree	String	Defines the name of the hierachical output of all data. If set to an empty string, no hierarchical output is used. Default: `'trees.txt'`
Out_JSON	String	Defines the name of the JSON ouput file. If set to an empty string, no JSON output is used. Default: `'json_dump.txt'`
Out_Tmp	String	Defines the name of the temporary storage. If the coding process gets terminated, all data is stored in this file. Currently there is no possibility to recover the coding automatically. Angrist 1.2 will be able to do so.
First_Page	String	Defines the page from which the coding process should be started.

The default value for each of these settings is set at the bottom of the script. If you wish to change the default values you may either change them at the bottom of the script (for all coders and once per session) within the function `self.fuellen()` (for all coders, set back to default for each text), or by using the settings file (see chapter 3.3).

If you are not sure whether a parameter has been set in the settings dictionary you may check using the function `available(setting)` where `setting` is a string variable with the name of the setting you wish to find. If there is a valid value in the settings dictionary for this key, the function returns `True`. If there is no valid setting, the function returns `False`.

4.2. Storage

`storage` is a global variable of the type dictionary. It contains all stored data for the current text. If there are multiple levels of analysis, the dictionary contains dictionaries for each unit of analysis. The format of values stored in this dictionary depend on the question type used for entering the values.

- For dropdown and radiobutton questions, storage contains a tuple with two elements (Label,Code) as string variables as value.
- For text questions, storage contains a string variable as value.
- For checkbox, rating, and sd questions, storage contains a dictionary with all options as keys and the entered values as values. For rating scales, the values may be defined manually. For checkbox and sd questions, the values are integers.
- For List questions, storage contains a tuple of lists ([Label1,Label2...],[Code1, Code2..]) which contains all labels and codes selected. If no multiple selection is allowed, the lists only contain one element.

Below you see a very brief example of the coding of a text using two levels of analysis (Text and Speaker) with some variables to code on each level. There are two units of analysis on the level of Speaker (S1 and S2), each with their own storage dictionary. On the level of text, the ID and length of the text were coded using a text question. The variables Genre and Author were coded using a dropdown menu. Framing was coded using a checkbox question with four options. On the level of speaker, three variables (ID, Position and Style) were coded using dropdown questions.

On each level there are two reserved variables which are coded automatically. `#TN` is the trivial name of the current unit of analysis. It is set in the function `self.level_down()` and is used for the review of coded units and informations on the current location. Its value may be changed at any time. The second reserved variable is `#TS`, the time stamp of the current unit of analysis. It marks the second the coder started coding this unit.

```
{
#TS:1381235730.75
ID:'Text1'
Genre:('Report','1')
Author:('Journalist','1')
Length:'300'
Speaker: {
    S1: {
        #TN:'Barrack Obama'
        #TS:1381235916.45
        ID:('Barrack Obama','1034')
        Presentation:('Positive','1')
        Voice:('Yes, direct quotation','2')}
    S2: {
        #TN:'Michelle Bachmann'
        #TS:1381235950.763
        ID:('Michelle Bachmann','1042')
        Presentation:('Negative','-1')
        Voice:('No','0')}
Framing: {
    Conflict:1
    Horse-Race:0
    Stratgy:0
    Attack:1}
}
```

Any entry in the storage may be assessed at any time in the program. If you wish to extract the code of the position of the first speaker, for example, you might call:

```
position = storage['Speaker']['S1']['Position'][1]
```

The value `'1'` would then be stored in the variable `position`.

When you are at a lower level of analysis and require a value coded within this unit of analysis, it may be doleful to enter the whole path to the current position manually. There is a shortcut you might use in this situation: If you call the function `curr()`, it returns the current sub-directory of storage you are editing. So, if you are still coding the first speaker and wish to know his position, you may call it using the command:


```
position = curr()['Position'][1]
```

Again, the value `'1'` is returned. This function is especially useful for analyses with more than two levels of analysis.

4.3. Codebook

codebook is a global variable of the type dictionary. It contains all variables of the codebook (usually a_codebuch.ini) as keys and the entries as value. The codebook dictionary is used in the presentation of question types and in the storage of variables. You may change the contents of the codebook at any time of the coding process.

The format of each entry in the codebook dictionary is a list containing five elements. Element 0 is the question asked to the coder as a string variable ending with a line-break. Element 1 is the coder information as a string variable. Element 2 is the list of labels for the options specified. Element 3 is the list of codes for the options specified. Element 4 is the help text offered to the coders.

Accordingly, the codebook entry for the example variable 'Author' (see Chapter 2.1.2) would be:

```
['Who is the author of this text?\n','Please indicate the author',['Journalist
      of the Medium','News Agency','Reader / Audience','National political
      actor','Other','Not identifiable'],['1','2','3','4','5','6'],'The author
      of a text refers to the person respo...']
```

When changing the entries of the codebook, these changes will persist for the whole coding session. They will, however, not be stored to the codebook file at the end of the text. The codebook is reloaded for each text in the coding session.

4.4. def_val

`def_val` is a global dictionary containing default values for certain variables. The keys of this dictionary are the names of the variables, the values are the values which should be set as default. You may set default values at any time of the coding process. If default values are set for a variable, the corresponding option is automatically selected as soon as the question for this variable is asked.

You may, for example, set the default value for the author of the text to 'Journalist', which has the code '1'. For this purpose you may use any of the following notations. All of them will work:

```
def_val['Author']='Journalist'
def_val['Author']='1'
def_val['Author']=('Journalist','1')
```

Setting default values is especially useful when using text- or data-mining techniques to predict the values of variables. It is also used to present the coder with the predictions of the automated content analysis using Aeglos (see Chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**).

4.5. prog_pos

`prog_pos` is a global variable of the type string. This variable contains the name of the current page the coder is presented with. The value of prog_pos is usually set and changed in the Submit-function after storing the values and before calling the Ask-function again.

Hint: It is strongly recommended to use informative names for the pages of your questionnaire. Names such as `'page1'`, `'page2'`, `'page2.2'` will make setting up of your program a nuisance. If you use `'author'`, `'topic'`, `'framing'` instead you will always know where you are in your script.

4.6. dta_pos

`dta_pos` is a global variable of the type list of strings. The value of this variable specifies the position within the data the coder is currently storing values to. The list has a fixed format:

```
['Level1','Unit1','Level2','Unit2','Level3','Unit3'...]
```

This list always contains at least four items. If the coder is at the top level of analysis, the value of the variable is `['-','-','-','-']`. When descending to lower levels, the dashes are replaced by the respective values. When coding the first speaker in the example above, the data position has the value `['Speaker','S1','-','-']`.

The value of the dta_pos variable is changed automatically by the `level_down()` and `level_up()` functions. You may change the value manually at any time. When doing this, be careful to match the entries in `dta_pos` to the currently existing dictionaries in `storage`.

5. Setting up Angrist

To demonstrate the process of defining a new project in Angrist, a very superficial and small example project is presented here. Parts of this example have been used in previous chapters. Here, we will set up the tool as a whole.

The aim of this example is to investigate the presentation of political actors in an American newspaper. For this purpose, the Genre of each article, its length and framing have to be recorded. In addition, all political actors appearing within the article shall be recorded and their presentation shall be assessed using two categories: Quotations and tone of presentation.

5.1. Making a plan

The first step when defining a new project should consist in a detailed plan of the pages of the questionnaire which are displayed in the tool. This plan may be used to subsequently set up the codebook, the Ask-Function, the Submit-Function, the Back-Function and other project specific parameters.

The plan should contain the following information:

- The names of the pages
- The names of the variables
- Question types
- Changes of the level of analysis

Table 1 shows an example of such a plan containing all relevant information.

Page	Questions: Variable and type	Actions upon submission
article properties	Author (dropdown) Genre (dropdown) Length (text 1 line)	
actor selection	Act_ID (dropdown)	If an actor is selected, change the level of analysis to Actor. Else go to page: framing
actor properties	Presentation Voice	Go back to actor selection and change to the top level of analysis
framing	Framing (rating scale 2 points)	
otherart	Otherart (dropdown)	If yes, restart angrist, if no, end angrist.

Table 1: Detailed plan for the example analysis

5.2. Definition of the codebook

First and foremost, the variables needed in this analysis have to be defined in a codebook. For this purpose, an empty textfile is created and stored as `a_codebook.ini`. This textfile is then filled with entries for the variables specified in the plan. As described in chapter , each entry begins with the name of the variable in brackets which is followed by three lines of information and a list of options.

For the variables used in this examples, the codebook would look like that:

```
############### Article Level

 [Author]
Who is the author of this text?
Please indicate the author
The author of a text refers to the person responsible for the content and presentation. In
       most cases the author is a journalist of the source in which the text has been
       published or a journalist of a news agency. In some cases, however, the source may
       be a private person, a politician, a reader (letters to the editor) or not
       mentioned at all. Since the names of the author is of no interest to the
       investigation of populism within the media, except if the person is a political
       actor, only the function of the author will be coded.
1:Journalist of the Medium
```

```
2:News Agency
3:Reader / Audience
4:National political actor
5:Other
6:Not identifiable

[Genre]
Type of the text
Which genre/category does this text belong to?
News story#Fact-oriented type of story, factual news report, report of events etc., of
       what has happened [when, where, who, what, why?], e.g., party meeting, report on
       recent events etc. Probably the most frequent type of news
       story.##Editorial/Column/Commentary/Letter to the editor#Most subjective kind of a
       news item in which authors give their personal interpretations and opinions:
       #Editorial: Typically explicitly marked as editorial, opinion-piece, an article of
       its own, clearly defined to give evaluations, typically on same page within
       newspaper each time. It has to be formally distinct from the rest of the page. It
       clearly expresses a standpoint of the author/editor who again speaks for his
       newspaper. #Column: clearly marked as special column, distinct from regular
       coverage, most likely always at the same place within news-paper, reoccurring item
       on a regular basis as fixed part of newspaper coverage, can be written in very
       personal style. #Commentary: Often not written by a journalist but by an external
       source such as an expert, politician etc.; often explicitly marked as „commentary“,
       e.g. by guest author. Letter to the editor: written by a reader; referring to
       previous articles published in the newspaper and representing the personal opinion
       of the reader; mostly explicitly marked as letter to the editor and placed on a
       special page within the newspaper.##Interview#News item consisting of questions and
       answers shown or printed in direct quotes between interviewer and interviewee or
       between several discussants. Note: There have to be at least two interview
       questions (often in bold or italic) to justify coding an interview! Interview
       sections which are part of a „reportage“ or another more subjective report are not
       meant here.##Background report/Reportage/Portrait/Feature#More subjective reports
       in different forms: #Feature article: Vivid report of a correspondent, named as the
       author of the article. #A „reportage“ describes individual experience of the
       author; often explicitly marked as „reportage“.#Background story: often longer
       article, not only factual reporting, looking behind the scenes, analytical, in-
       depth - not only descriptive, often explicitly marked as „analysis“, etc. Magazine
       style report: Type of news story in which elements of factual news reports and
       subjective elements intermingle. But: The journalist does not have to be on
       location, describing individual experience as in a reportage/background
       story.#Portrait: Portrait of, e.g., a person, group, institution, organization -
       and nothing more than that. Otherwise it may be a news story or a reportage (see
       above).
1:News Story
2:Editorial/Column/Commentary/Letter to the Editor
3:Interview
4:Background Report/Reportage/Portrait/Feature

[Length]
Length of the text
Please enter the number of words below.
The number of words may be found at the top of the text next to the word 'LENGTH:'

[Framing]
How is the text framed?
Please check the presence and absence of these frames
Conflict-Framing:#There is a conflict between two or more actors. There is reproach, a
       winner, or a loser.##Horse-Race Framing:#Candidates are presented in a sports-like
       fashion and their chances to win or to lose are caluclated. There are points,
       percentages, or polls to support the prediction.##Strategy-Framing:#Politicians are
       portrayed as schemers making plans, intrigues, tactics, or strategies to reach
       their goals. Politics in general is portrayed as a strategy game.##Attack-
       Framing:#Direct attacks of actors against others are portrayed or the journalist is
       directly attacking a politician.
Conflict:Conflict-Framing
Horse-Race:Horse Race-Framing
Strategy:Strategy-Framing
Attack:Attack-Framing


[Otherart]
Do you want to code another article?
All entries you made have been stored. You may now choose to continue with another
       article.
-
1:Yes
2:No
```



```
############### Actor Level

[Act_ID]
Which actor is it?
Please select the appropriate actor from the list below.
You may enter some characters of the name of this actor to restrict the list.
x:No more actors
1:Boehner, John (Rep., R.)
2:Cruz, Ted (Sen. R.)
3:Gray, Vincent C. (Mayor Washington DC)
4:Holmes Norton, Eleanor (Del. D.)
5:Issa, Darrell (Rep. R.)
6:Needham, Michael
7:Obama, Barrack (Pres. D.)
8:Reid, Harry (Sen. D.)
9:other actor

[Presentation]
How is the actor presented?
Please select
The presentation refers both to verbal presentation and pictures.
1:Positive
0:Ambiguous/Neutral
-1:Negative

[Voice]
Is the actor quoted in this article?
Please select
A direct quote is always put in quotation marks. Indirect quotes may be recognized by the
0:No
1:Yes, indirect quote
2:Yes, direct quote
```

Please do not bother to take the content of this codebook seriously. It was created for purely formal reasons and does not comply with standards of quantitative content analysis.

5.3. fuellen()

With the codebook defined we may now turn to setting up the script angrist.py. For this purpose, a series of functions has to be defined and customized.

The first of these functions is `fuellen()` which sets initial values for the settings, storage and codebook directories and defines global variables. While most of this function should not be modified, you might want to change some of the settings for this content analysis.

Since it would be nice to automatically highlight the names of the actors we code in this analysis, we might insert the following lines:

```
settings['Hotwords']={'us':['Boehner','Cruz','Gray','Holmes','Norton','Issa',
                                        'Needham','Obama','Reid']}
settings['Country']='us'
settings['Auto_Highlight']='1'
```

These lines set the country of the content analysis to `'us'` and cause the GUI to highlight a list of words upon loading. The words are case sensitive but may occur at any place within a word.

In addition to these settings we want to start the content analysis on page 'article properties' and enable the coder to highlight found actors using a yellow marker. Finally, one of the variables is entered by means of a rating question and will need more space in the data and we want to reset the list of actors for each text.

```
settings['Highlight_Buttons'] = [['Actor','Act','#ffff66']]
settings['First_Page'] = 'article properties'

settings['Multi_Items'] = ['Framing']
if 'Actor' in settings.keys():
    del settings['Actor'] ##Reset upon loading new text
```

Except from these additions, the `fuellen()`-function may be left unchanged for this project.

5.4. ask()

The Ask-Function contains all information for the display of questions. For each page, a series of commands may be specified, which call questions and the buttons at the bottom of the page. The commands are embedded in a large `if..elif..else`-block in which a command block is defined for each possible value of the variable `prog_pos`.

In this project we shall need five of these blocks which are called as specified in the plan above. In each block the questions for the variables on this page may be called using `self.question`-functions. In addition, a review of previously coded actors may be displayed at the page `'actor selection'`.

```
if prog_pos == 'article properties': #First page if no ID was found in any to-do
       list
    self.question_dd('Author',1)
    self.question_dd('Genre',2)
    self.question_txt('Length',3)

elif prog_pos == 'actor selection':
    self.show_review('Actor',1)
    self.question_dd('Act_ID',1)

elif prog_pos == 'actor properties':
    self.question_dd('Presentation',1)
    self.question_dd('Voice',2)

elif prog_pos == 'framing':
    self.question_rating('Framing',1,['present','absent'],['1','0'],defval='0')

elif prog_pos == 'otherart':
    self.buttons(0,0,0,1)
    self.ausblenden()

    check_todo()

    self.question_bt('Otherart',1)
    self.f_questions.bu1_1["command"] = self.submit
    self.f_questions.bu1_2["command"] = self.abort
    self.f_questions.bu1_1.bind('<Return>',self.submit)
    self.f_questions.bu1_1.focus()
```

Since the text is assessed completely by the time the coder reaches the page 'otherart', the current ID may be checked off on the todo list. To do so, the function `check_todo()` is called. This function removes the current ID from the todo list. Also, at the end of a text, the text display may be removed. This is achieved by calling `self.ausblenden()`. Afterwards, the question whether another article should be coded, is displayed to the coder. Since the question is a buttons question, the check button is suppressed on this page by means of the function `self.buttons()`.

5.4.1. Data storage

The code above would ask the coder for all entries but it would not store any information to external files. All information entered by the coder would be lost after finishing the text. Data may be stored using built-in functions which may store the data as tree hierarchy, as JSON output or as relational tables.

For all three ways of data storage, there is a function. Since the functions need a setting defining the name of the output-file it pays only to call them if the setting is available. You may check availability of any setting by calling `available(setting)`.

```
baum_export()

self.export_data([],['#TS','Author','Genre',
                     'Length','Framing'],'_Text.txt')
```



```
self.export_data(['Actor'],['#TS','Act_ID','Presentation',
                            'Voice'],'_Actor.txt')
```

These commands produce output in four different files. First, a tree representation of the storage dictionary is stored. Depending on the values of `settings['Out_Tree']` and `settings['Out_JSON']` an indented tree or a JSON-Output or both is written to a file. Second, all data on text level is exported to a table containing the columns `'#TS'`, `'Author'`, `'Genre'`, `'Length'`, and four dummy variables for the variable `'Framing'`. Finally, for all elements on level `'Actor'`, a case is exported to a table containing the columns `'#TS'`, `'Act_ID'`, `'Presentation'`, and `'Voice'`. In addition, each case will contain information on the coder, the text ID and the duration of the coding.

5.4.2. Calling functions using button questions

In this example there is only one button question at the end of the questionnaire. Variable `Otherart` may be answered with yes or no, leading to different paths in the coding process. For these decision posints, button questions may be used.

Button questions generate up to four buttons from options provided in the codebook. In this case, a button named 'Yes' and a button named 'No' is generated. These buttons are void of any function and have to be defined in order to work.

There are two different possibilities for linking these buttons to functions. First, you may bind the buttons to different existing or custom functions in which commands for the current program position are defined. This is what is done in the example above by linking the first button to self.submit() and the second button to self.abort().

Another way to bind buttons to functions is binding all buttons to the submit function with unique parameters. The function submit() takes one argument if necessary. This argument may be used to identify the button pressed. To submit an argument to the Submit-function, the CMD-class has to be used as no parameters are usually allowed when binding a command to a button:

```
self.question_bt('Otherart',1)
self.f_questions.bu1_1['command'] = CMD(self.submit,1)
self.f_questions.bu1_2['command'] = CMD(self.submit,2)
```

This code links both buttons to the Submit-function but it will set the value of the parameter `overspill` to `1` or `2`. This value may be used in deciding which action to take.

5.5. submit()

The Submit-function is called when pressing the check-button. It handles the internal storage of items, the cleaning up of pages and the redirection to subsequent pages. In the plan, three exceptional actions have been noted. When storing the entry of the page `'actor selection'`, the level of analysis has to be changed to `'Actor'`. Then storing the entries of the page `'actor properties'`, the level of analysis has to be set back to the top level. When storing the entry of `'otherart'` (in the definition of this page, the first button of the buttons question has been linked to `self.submit()`, so the coder wants to code another text), the GUI is reset by calling `self.fuellen()`.

The Submit-function always checks the entries before doing anything else. This is done by calling the function `self.check_entries()`. If the check fails, an error message will be produced and the entries will not be stored or processed.

5.5.1. Storing values

In the Submit-function, entries by the coder may be stored to the internal storage dictionary. To do so, you may use the function `self.store_var(var)` or `self.store_var_all()`. Both functions store the

entered value of one or all variables on the page to the respective key in the storage dictionary at the data position indicated by `dta_pos`. If you want to store the values to some other key within this dictionary you may do so manually.

For the page 'text properties', all three versions of storage commands shown below do exactly the same thing:

```
if prog_pos == 'article properties':
    self.store_var_all()

if prog_pos == 'article properties':
    self.store_var('Genre')
    self.store_var('Author')
    self.store_var('Length')

if prog_pos == 'article properties':
    storage['Genre'] = self.store_var('Genre',store=0)
    storage['Author'] = self.store_var('Author',store=0)
    storage['Length'] = self.store_var('Length',store=0)
```

The parameter store=0 in the function call of self.store_var() causes the value not to be stored automatically but to be returned in order to store it manually. In this case, the result is the same. You may, however, choose to look at a value without storing it yet. In these cases, the option is used.

5.5.2. Changing the level of analysis

There are different ways to change the level of analysis. The most simple one consists in calling the function `self.level_down(var,level)`, where `var` denominates the variable containing information on the identity of the unit and `level` denominates the level on which the unit of analysis is coded. Conversely, the function `self.level_up()` may be called to return to the level above the current one.

There is another way to change the level of analysis which is not used in this example. There, the coder is prompted to highlight all units of analysis within the text. The highlighted portions of the text are counted and will be presented to the coder who may then code them subsequently. To use this feature, an additional page has to be included and the coder has to have a marker defined in `settings['Higlight_Buttons']`.

On a first page, the coder is prompted to mark all units of analysis by using the question `self.question_mark_units(var,Level)`. The highlighted portions may be stored in the Submit-function by calling `self.unit_confirm(Level)`. Subsequently, by using the question function `self.question_sel_units(var,Level)`, in the Ask-function, the coder is prompted to select the next unit of analysis. Finally, the selection may be used to change the level of analysis by calling the function `self.unit_select(Level)` in the Submit-function. For all functions, the value Level refers to the level of analysis which shall be used to store the units. `var` denominates the codebook variable containing the question- and help-text for the questions.

In this example the handshake between these functions would look like this:

Ask-Function:

```
elif prog_pos == 'mark actors':
    self.buttons(0,0,0,0)
    self.question_mark_units('Mark_Act','Actor')

elif prog_pos == 'select actor':
    self.buttons(1,0,0,1)
    self.clean_all_tags() #clean up highlights
    self.show_review('Actor',1) #show previously coded units
    self.question_sel_units('Sel_Act','Actor')
```

Submit-Function:


```
elif prog_pos == 'mark actors':
    self.clean_up_all()
    self.unit_confirm('Actor')
    prog_pos = 'select actor'
    self.ask()

elif prog_pos == 'select actor':
    self.clean_all_tags()
    self.unit_select('Actor')
    self.clean_up_all()
    self.ask()
```

Evidently, two additional variables have to be defined in the codebook for this to work.

When creating a new unit of analysis using this method, the position of the highlighted text, its content and the content of the whole paragraph are stored in a dictionary in settings. There, too, a value is stored indicating whether this unit of analysis has been coded already or has to be shown to the coder again. In addition, the type of the marker and a label for list display are stored.

Key	Wert
Start	Position of the first character of the selection in the text
End	Position of the last character of the selection in the text
Fulltext	Text of the whole paragraph
Wording	Text of the highlighted portion of text
Done	Integer with value 0 for uncoded units and value 1 for coded units of analysis.
Label	Label of the unit of analysis to be displayed in lists. Usually shorter than 40 characters.
Typ	Type of the marker used to highlight this unit of analysis as defined in `settings['Higlight_Buttons']`

Table 2: Keys and content of the dictionary created at settings[Level][Unit] for each highlighted portion of the text.

5.5.3. Cleaning up the page

The page may be cleaned from all questions using the function `self.clean_all()`. If only one question is to be removed while the other ones remain in place, you may also call `self.clean(var)` where `var` denominates the variable behind the question to be removed.

5.5.4. Complete submit function

Taken together, the submit function for the current program looks like that:

```
accept_entry = self.check_entries()
if accept_entry == 1:
    if prog_pos == 'article properties':
        self.store_var_all()
        self.clean_up_all()
        prog_pos = 'actor selection'
        self.ask()

    elif prog_pos == 'actor selection':
        act = self.store_var('Act_ID',store=0)[1]
        if act == 'x':
            self.clean_up_all()
            prog_pos = 'framing'
            self.ask()
        else:
            if self.level_down('Act_ID','Actor'):
                self.store_var_all()
                self.clean_up_all()
                prog_pos = 'actor properties'
                self.ask()

    elif prog_pos == 'actor properties':
        self.store_var_all()
        self.clean_up_all()
        self.level_up()
```

```
        prog_pos = 'actor selection'
        self.ask()

    elif prog_pos == 'framing':
        self.store_var_all()
        self.clean_up_all()
        prog_pos = 'otherart'
        self.ask()

    elif prog_pos == 'otherart':
        self.fuellen()

    else:
        if settings['Verbose'] == '1':
            verb("Error: Position '"+prog_pos+"' is not defined for SUBMIT")
else:
    verb('Invalid Entries')
```

Within the command blocks, the storage function has to be called before the cleaning function. Upon cleaning, most entries are removed from the cache and the entries may not be stored anymore.

All commands within the Submit-function have to end with calling `self.ask()` again. If this function is not called, the GUI stalls and does not proceed with data input. If the value of `prog_pos` is not changed before calling the Ask-function, the GUI enters an endless loop.

On page `'actor selection'` the value of variable `'Act_ID'` is used to decide which path to take. If the selection is `'No more actors'` which has code `'x'`, the next page is `'framing'`. For all other values of `'Act_ID'` the level of analysis is changed. This is done using an `if..else` statement.

At some points of the Submit-function, the function `verb()` is called. This function takes one string as argument and writes the string to a log. The log may be seen when debugging but remains hidden from the coder.

5.6. abort()

The function self.abort() will be called if the abort button is pressed by the coder or if a buttons question links a button to this function. In this example, the abort button is not displayed on any page. There is, however a link to the function in the definition of the buttons question on page `'otherart'`. This means that the Abort-function has to be defined for this page:

```
        if prog_pos == ['otherart']:
            self.clean_up_all()
            prog_pos = 'ende'
            self.ask()
```

In this case, the page is cleaned up and the coder is redirected to the page 'ende' which informs him that all data has been stored and the window may be closed. Such a definition has to be done for any page where a button calls the function `self.abort()`.

5.7. back()

The function self.back() is called whenever the coder clicks the Back-button at the bottom of the page. This button is displayed by default and has to be disabled manually for pages where it makes no sense.

The Back-function usually sets the value of prog_pos to the last page visited, cleans up the page and calls self.ask() again. If any other actions should be taken when pressing 'Back' on a page, commands have to be defined manually.

In this example, special precautions have only to be taken for one page. When going back from the page 'actor properties', the level of analysis has to be set back to the top-level. This may be done using the following lines:


```
if prog_pos == ['actor properties']:
    del storage[dta_pos[0]][dta_pos[1]]
    self.level_up()
    prog_pos = 'actor selection'
```

The function `self.ask()` will be called at any rate so it does not have to be called in the if..else block of this function.

5.8. rb_tamper()

The function `self.rb_tamper()` is called whenever the value of a radiobuttons question is changed by the coder. The function may be used to change the appearance of the current page according to the entry of the coder.

As in all other functions in angrist, `self.rb_tamper()` contains an `if..elif..else`-block in which you may define commands for any page. Since no radiobutton is used in the example above, another example is shown here to demonstrate the definition of this function.

You may chose to ask for the exact news agency if the coder selects `'News Agency'` (code 2) as author of the text. If the variable 'Author' is answered by radiobuttons. Imagine there is another variable 'NewsAgency' which contains all news agencies as options:

```
if prog_pos=='article_properties':
    self.clean_up('dd',3)
    if storage['Author'][1] == '2':
        self.question_dd('NewsAgency',3)
```

These lines first remove the question at position 3 (if available). Then, if the value of `'Author'` is `'2'`, variable `'NewsAgency'` is asked using a dropdown question.

With the definition of these functions, the program runs perfectly well. The whole example as a working angrist project may be downloaded from the resources page at:

http://www.ipmz.uzh.ch/Abteilungen/Medienpsychologie/Recource/Angrist.html

6. Extension: Aeglos

As a python script, Angrist may be extended using a wide range of packeages for statistical processing, natrual language recognition, or data management which are available for python today. There is, however, one package which is designed explictly for the use with Angrist and may be used for semi-automated content analysis.

Semi-automated content analysis (SACA) is a method in which techniques for automated content analysis are linked to manual data entry with the purpose of enhancing the efficiency of the coders (Wettstein, In Press). A central element of SACA is the prediction of values the coder is likely to enter in order to preselect them in the query form. This has been found to have two different effects on the coder. First, the coder does not have to bother clicking the obvious choice from a dropdown which saves time and trouble in the content analysis. Second, the task of the coder changes from assessing the text on his own to checking and correcting the program's analysis. In cases where the program does not select the option which would be preferred by the coder, he reacts with increased elaboration of the text and pays higher attention to the semantic meaning of the text (Wettstein, in Press).

The Angrist Extension Generating Listselections for Optimal Speed (Aeglos) is a module enabling Angrist to learn from previous decisions of the coder and calculate the probability for the selection of any option for any variable in the codebook for unknown texts. The technique used for these predictions is a Naive Bayes Classifier (cf. Manning/Schütze...) which uses the conditional probabilities for words to occur in texts in which an option has been chosen to predict the probability of this option to be chosen in a new text for which only the words are known. Aeglos may thereby be trained directly from the data output of Angrist and is used in all question types to generate default values when it is activated.

This part of the documentation is being developed and will be included by summer 2015.

7. Glossary

Angrist uses a small number of predefined functions which may be used in the project definition. Some of these functions have been mentioned or explained using the example project above. In this glossary, all functions are presented for a general overview. The functions are grouped and sorted by type and usage.

7.1. Initialization

`self.fuellen()`

This function initiates the GUI and sets the basic settings for each text. This includes the coder specific settings (by referring to `a_settings.ini`), the codebook (by referring to `a_codebook.ini`), the ID of the text (by referring to the todo-list) and manually defined settings which are used in the content analysis.

When this function is called, the content analysis of the current text is set back to zero. All entries are deleted and the content analysis starts from the initial page. Therefore, it is important not to call this function in the program without cautioning the coder and saving all data.

For the correct usage of `self.fuellen()`, refer to chapter 5.3.

`self.set_window()`

This function is called at the beginning of each session to set up the layout of the GUI. Any changes made in this function will affect the layout of all pages in the questionnaire.

For a detailed description of the layout, see chapter 2.

7.2. Project specific functions

`self.ask()`

This function is used to define the pages which are shown to the coder. The function contains a large `if..elif..else`-block in which a series of commands is defined for each value of `prog_pos`. Each block of commands usually contains a `self.buttons()`-command to define the buttons displayed on the page and a series of `self.question_XX()`-commands to call the appropriate question type for each variable.

For a detailed description of this function, refer to chapter 5.4.

`self.submit(overspill=0)`

This function is called whenever the coder hits the Check-button and thus submits the decisions to the program. Just like `self.ask()`, the Submit-Function contains a large `if..elif..else`-block defining the actions to be taken for each value of `prog_pos`.

The actions usually include storing the values to the internal storage, cleaning up the page, setting a new value for `prog_pos`, and calling the Ask-function again.

The Submit-function may take one argument. This is necessary for binding the function to events in tkinter. The argument may also be used to submit additional information to the function such as the identity of the button pressed.

For a detailed description of this function, refer to chapter 5.5.

self.abort()

This function is called by clicks on the Abort-button or by answers to buttons questions, which are linked to this function. Again, this function is defined for each value of `prog_pos` by means of an `if..elif..else`-block.

For a detailed description of this function, refer to chapter 5.6.

self.back()

This function is called by clicks on the Back-button. Its general setup sets back the program position by one page and calls the Ask-function again. If any other action is necessary to go back from a certain positon within the program, this action has to be defined using the `if..elif..else`-block.

For a detailed description of this function, refer to chapter 5.7.

self.rb_tamper()

This function is called by changing the value of any radiobutton question in the questionnaire. The actions to be taken when the value of a question is changed have to be defined for the respective page by means of an `if..elif..else`-block.

7.3. Important aides

This set of functions may be called in the Ask- or Submit-function to change the appearance of the GUI or to manipulate the data. The definition of these functions should only be tampered with by people with good knowledge of Python.

self.show_review(level,rm=1,edit=0,height=3)

This function displays a list of all items on a given level below the current level of analysis. This review list is displayed above the current page and there is a possibility to change the entries either by removing them from storage or editing them.

All entries in this list have the form '<*ID*>: *Trivialname*' where *ID* is the identifier of the unit of analysis for this entry and *Trivialname* is the value of the key `'#TN'` of this unit of analysis. The values for `'#TN'` are set automatically when using the `level_down()` function. They may then be changed manually if desired.

Options:

`level`	Either a string variable denominating the level of analysis or a list of strings denominating multiple levels of analysis. The string variables must be identical with the name of the level of analysis within the `storage` dictionary.
`rm`	Integer variable which may take the values 1 or 0. If set to 1, a remove-button is displayed next to the list which allows the coder to remove a unit of analysis from storage. Default: 1
`edit`	Integer variable which may take the values 1 or 0. If set to 1, a edit-button is displayed next to the list which allows the coder to edit a unit of analysis from storage. Default: 0 In order for this button to work, the function `self.edit_item()` has to be defined for the current project.
`height`	Height of the listbox in lines. Default: 3

Example:

```
self.review(['Actor','Issue'],1,0)
```


This example displays a list of all units of analysis that have been coded on the levels `'Actor'` and `'Issue'` which are below the current level of analysis. The coder may remove any item from the list but is not allowed to edit them.

self.hide_review()

This function removes the review list from the top of the current page. By default, this function is called for all pages defined in self.ask(). If a review list is desired it has to be called for each page where it is needed.

self.remove_item(level, height)

This function is called by pressing the remove button next to the review list. The parameters `level` and `height` are taken from the `self.show_review()` function and are used to display the list after removing an item. The function removes the selected item from storage and redisplays the list.

self.edit_item(level)

This function is called by pressing the edit button next to the review list. The parameter `level` is taken from the `self.show_review()` function. The function sets the data position to the selected unit of analysis and the program position to a page specified in the project. For each project this function has to be customized for the button to redirect the coder to the correct page within the questionnaire.

self.level_up()

This function changes the level of analysis to the parent level.

self.level_down(variable,level)

This function changes the level of analysis to a lower level. For this purpose, the values entered for a variable are used. This function is usually called in the Submit-function in order to change the unit of analysis according to coder inputs.

The function returns an integer with value 1 if the new unit of analysis was created successfully.

Options:

`variable`	String variable denominating the variable which is to be used to name the unit of analysis. The code of the value entered is used as internal identifier of the unit of analysis. The label of the option is used as trivialname and is stored in curr()['#TN'] for usage in other functions.
`level`	String variable denominating the level on which the unit of analysis is to be created.

Example:

When on text level and choosing actor 'Barrack Obama' with code 'us01' in variable 'Actor_ID' as actor to be coded in a new unit of analysis, the command:

```
self.level_down('Actor_ID','Actor')
```

Changes the value of `dta_pos` to `['Actor','us01']` and sets the value of `storage['Actor']['us01']['#TN']` to `'Barrack Obama'`. Accordingly, all future entries will be stored in this unit of analysis until the command `self.level_up()` sets the level of analysis back to text level.

self.buttons(check=1,abort=0,back=1,pause=1)

This function displays buttons below the current page of the questionnaire. If the function is called without any arguments, all buttons except for the Abort-Button are displayed. See also: chapter 2.

Options:

`check`	Integer variable with value 0 or 1. If set to 1, the Check-button is displayed. Default: 1
`abort`	Integer variable with value 0 or 1. If set to 1, the Abort-button is displayed. Default: 0
`back`	Integer variable with value 0 or 1. If set to 1, the Back-button is displayed. Default: 1
`pause`	Integer variable with value 0 or 1. If set to 1, the Break-button is displayed. Default: 1

Examples:

```
self.buttons(1,1,0,0)
```

Displays the Check- and Abort-buttons but suppresses Break and Back.

```
self.buttons(abort=1)
```

Displays all four buttons.

self.check_entries()

This function checks all answers for validity. The function returns `1` if all entries are valid and `0` if they are not. This function is called whenever the Submit-function is called. All commands in the Submit-function are only executed if `self.check_entries()` returns `1`.

The function returns 0 if:

- A dropdown or radiobutton question is answered with an option that has code '98'
- Text has been entered to a textbox which may not be processed

The function returns 1 if no error occurred or if `settings['Debugging']` is set to `'1'`.

self.clean_up(pos, typ='')

This function removes a question of type typ from position pos. If no value for parameter typ is provided, the function removes any question at this position. If there is no question with type typ at the given position, no action is taken.

Options:

`pos`	Integer variable with value 1, 2, or 3. Denominates the position of the question to remove.
`typ`	String variable naming the type of question to remove. Possible values: `'dd'`: dropdown question `'txt'`/`'txt2'`: text question `'rb'`: radiobutton question `'list'`: listbox `'listseek'`: searchable list `'listadd'`: list selection `'unit_auswahl'`: unit selection

`'rb'`/`'sd'`/`'rating'`/`'cb'`:	radiobutton, sd, rating, or checbox question
`'bt'`:	buttons question

self.clean_up_all()

This function cleans up the whole page, removing all questions by calling `self.clean_up()` for all positions.

self.codegetter(variabel, item)

This function looks up the code for an option for which only the name is known. It returns the code as a string variable.

Options:

`variable`	String variable denominating the variable which has the demanded option.
`item`	String variable with the exact label of the option which is demanded.

self.namegetter(variable, item)

This function looks up the label for an option for which only the code is known. It returns the label as a string variable.

Options:

`variable`	String variable denominating the variable which has the demanded option.
`item`	String variable with the exact code of the option which is demanded.

self.unit_select(level):

This function may be called to read the input of a `question_sel_units()` question and automatically open a new unit of analysis on the level specified. In addition, the function highlights the current unit of analysis in the text for better navigation.

The trivialname (`curr()['#TN']`) of this unit of analysis is set to the wording of the highlighted text.

Options:

`level`	String variable denominating the level on which the unit of analysis is to be created.

self.unit_confirm(level,sel_tags=[]):

This function is used to create a list of units of analysis from highlighted portions of the text. The function reads all marks in the text and translates them to potential units of analysis which may be displayed in a `question_sel_units()` question.

Options:

`level`	String variable denominating the level on which the unit of analysis is to be created.

`sel_tags` List of strings denominating the highlight-buttons which are to be read. If sel_tags is set to an empty list, all tags are converted to units of analysis. Default: `[]`

`self.store_var_all(setdef=0)`

This function stores all entries from the current page to the storage dictionary. All entries are stored at the position specified by dta_pos.

Options:

`setdef` Integer variable which may take the values 1 or 0. If set to 1, the current entries are set as default for future questions of the same variable. Default: `0`

`self.store_var(variable,pos=-1,setdef=0,store=1)`

This function stores the entry for a given variable to the storage dictionary. The function is usually called by self.store_var_all() but may also be used seperately in the Submit-function. The function returns the stored value.

Options:

`variable` String variable denominating the variable which is to be stored.

`pos` Position of the question on the current page.

`setdef` Integer variable which may take the values 1 or 0. If set to 1, the current entries are set as default for future questions of the same variable. Default: `0`

`store` Integer variable which may take the values 1 or 0. If set to 0, the value is not stored in `storage`. The function nevertheless returns the value. Default: `1`

The values returned and stored by this function depend on the type of question which was used to ask for the value:

- For dropdown and radiobutton questions, storage contains a tuple with two elements (Label,Code) as string variables as value.
- For text questions, storage contains a string variable as value.
- For checkbox, rating, and sd questions, storage contains a dictionary with all options as keys and the entered values as values. For rating scales, the values may be defined manually. For checkbox and sd questions, the values are integers.
- For List questions, storage contains a tuple of lists ([Label1,Label2...],[Code1, Code2..]) which contains all labels and codes selected. If no multiple selection is allowed, the lists only contain one element.

`self.message(m_id,m_type=1,variable='Err_Msg')`

This function displays a message of a given type to the coder. The message is displayed as a pop-up and may be closed by pressing 'OK' or closing the window.

Options:

`m_id` String variable denominating the code of the message to be shown. The code has to be defined in the codebook for the `variable` specified.

`m_type` Integer which may take the values 1, 2, or 3 and specifies the type of message to be displayed.
`1`: Warning to the coder.
`2`: Information to the coder.
`3`: OK/Cancel-question. If this type of message is used, the function returns 1 if the coder pressed 'OK' and 0 if the coder pressed 'Cancel'. You may use this type of message for critical decisions.

`variable` String variable denominating the codebook variable in which the message is defined as an option. Default: `'Err_Msg'`

In the codebook, a set of error messages is defined:

```
[Err_Msg]
-
-
-
Caution01:Warning!#If you go back one step further, the coding will terminate
      and the text will be loaded again. All entries you made will then be lost
      and the coding process starts fresh for this text.##Are you sure you want
      to go back?
Caution02:This action will remove the text from the sample. Only discard texts
      that do not have anything to do with elections, immigration, or labor
      market policies.##Are you sure you want to discard this text?
Info01:Your insecurity has been transmitted. You may now continue coding.
Invalid-Selection01:No Item was selected from the list. Please select at least
      one item to continue.
Invalid-Selection02:Invalid selection. Please select an option to
      continue.##Variable:
Invalid-Selection03:Invalid Characters entered. Please do not use special
      characters.
Runtime-Error01:This action is impossible.##Please contact Martin Wettstein to
      find out why.
Runtime-Error02:There are no more items in the Todo-List or the Todo-List does
      not exist. Please contact the operational staff to get new texts or fix
      this problem.##If you do know the ID of your next text you may enter it
      manually.
Runtime-Error03:Please enter the text identifier. You may not continue coding
      without a valid identifier.
Runtime-Error04:The specified text was not found in the text-folder. Please
      check spelling.
Runtime-Error05:Please select Source and proceed to the next page before using
      this feature.##The highlighting of relevant words will depend on the
      country of origin of your Medium
```

You may define any other message in this variable or create new variables with messages to be called.

7.4. Question functions

Question functions are usually called in the Ask-function and are used to prompt data input by the coder. There is a variety of different question types available.

`self.question_dd(var,pos,width=40)`

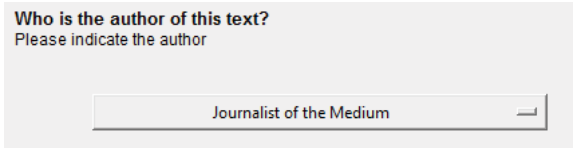
This function creates a dropdown question for variable `var` at position `pos`.

Options:

`var` String variable denominating the codebook variable to be asked.

`pos` Integer variable which may take the values 1, 2, or 3 and which indicates the position of the question on the page.

`width` Integer variable defining the width of the dropdown menu in characters. Default: `40`



`self.question_txt(var,pos,width=40)`

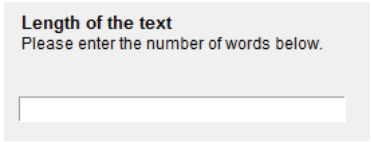
This function creates a textbox question for variable `var` at position `pos`.

Options:

`var` String variable denominating the codebook variable to be asked.

`pos` Integer variable which may take the values 1, 2, or 3 and which indicates the position of the question on the page.

`width` Integer variable defining the width of the textbox in characters. Default: `40`



`self.question_txt2(var,pos,width=40,height=3,getselect=0)`

This function creates a textbox question with multiple lines for variable `var` at position `pos`.

Options:

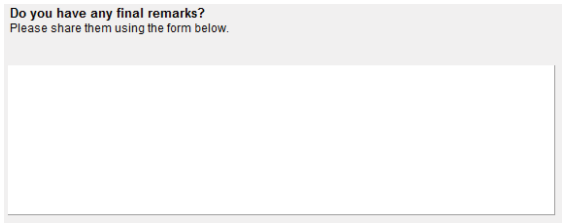
`var` String variable denominating the codebook variable to be asked.

`pos` Integer variable which may take the values 1, 2, or 3 and which indicates the position of the question on the page.

`width` Integer variable defining the width of the textbox in characters. Default: `40`

`height` Integer variable defining the height of the textbox in lines. Default: `3`

`getselect` Integer variable which may be 1 or 0. If set to 1, a button is created to copy any text selection within the text display to the question. This saves the coder a Ctrl+C/Ctrl+V. Default: `0`



Example: self.question_txt2('Remarks',3,width=80,height=10)

`self.question_cb(var,pos,layout='hor',defval=0)`

This function creates a checkbox question for variable `var` at position `pos`. The question takes up to 14 options to be checked.

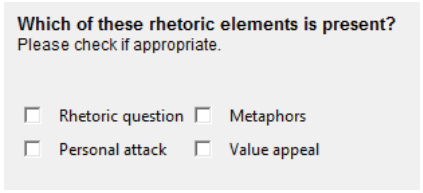
Options:

`var` String variable denominating the codebook variable to be asked.

`pos` Integer variable which may take the values 1, 2, or 3 and which indicates the position of the question on the page.

`layout` String variable which may take the values `'hor'` or `'vert'`. If set to `'hor'`, the options are presented in subsequent rows of 2 items each. If set to `'vert'`, the options are presented in subsequent columns of 7 items each. Default: `'hor'`

`defval` Default value for all checkboxes. If set to 1 all checkboxes are selected. If set to 0, none is selected. This default value is overruled by any previous coding or explicitly set default value in the `def_val` directory. Default: `0`



`self.question_rb(var,pos,layout='vert',defval='98')`

This function creates a radiobutton question for variable `var` at position `pos`. The question takes up to 14 options to be selected.

Options:

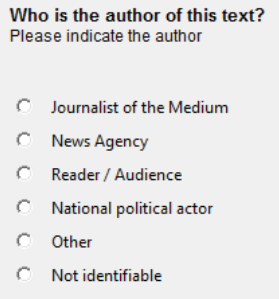
`var` String variable denominating the codebook variable to be asked.

`pos` Integer variable which may take the values 1, 2, or 3 and which indicates the position of the question on the page.

`layout` String variable which may take the values `'hor'` or `'vert'`. If set to `'hor'`, the options are presented in subsequent rows of 2 items each. If set to `'vert'`, the options are

presented in subsequent columns of 7 items each. Default: `'vert'`

`defval` Default value for the selection. This default value is overruled by any previous coding or explicitly set default value in the `def_val` directory. Default: `'98'`



`self.question_sd(var,pos,points=5,defval=0)`

This function creates a semantic differential question for variable `var` at position `pos`. The question takes up to 7 options to be rated on a scale between two extremes. The extreme values are the code and the label of each option of this variable. The code thereby represents the lowest rating (i.e. `0`), the label represents the highest rating.

Options:

`var` String variable denominating the codebook variable to be asked.

`pos` Integer variable which may take the values 1, 2, or 3 and which indicates the position of the question on the page.

`points` Integer variable defining the number of points which may be selected between the extremes. Default: 5

`defval` Default value for all ratings. This default value is overruled by any previous coding or explicitly set default value in the `def_val` directory. Default: `0`

`self.question_rating(var, pos, scalelist=['disagree','','','','agree'], valuelist=['1','2','3','4','5'], defval='1')`

This function creates a rating question for variable `var` at position `pos`. The question takes up to 7 options to be rated on a scale which may be defined freely.

Options:

`var` String variable denominating the codebook variable to be asked.

`pos` Integer variable which may take the values 1, 2, or 3 and which indicates the position of the question on the page.

`scalelist` List of strings which contains the labels of the rating scale. If a point does not have a

label, insert an empty string. Default: `['disagree','','','','agree']`

`valuelist` List of strings or integers which contains the values for each point which may be selected. The lenght of this list has to be identical to scalelist. Default: `['1','2','3','4','5']`

`defval` Default value for all ratings. This default value is overruled by any previous coding or explicitly set default value in the `def_val` directory. If `defval` is off the scale (i.e. not part of `valuelist`), no default value is set. Default: `'1'`

`self.question_bt(var,pos)`

This function creates a buttons question for variable `var` at position `pos`. The question takes up to four options to be displayed as buttons.

Options:

`var` String variable denominating the codebook variable to be asked.

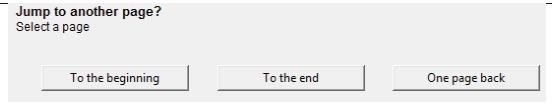
`pos` Integer variable which may take the values 1, 2, or 3 and which indicates the position of the question on the page.

When a button is pressed, the Submit Function is called with the parameter 'button' indicating the button which was pressed. You may use this information within the Submit-Function to decide upon further actions. Just use the variable `button` which is set to the value of the option which was pressed.

Example:

This short example demonstrates the use of three buttons to jump forward and back in the questionnaire. This function is unlikely to be of any use but it shows the handling of this question type.

Codebook	`[Jump]`
	`Select a page`
	`Jump to another page?`
	`No Helptext available`
	`start:To the beginning`
	`end:To the end`
	`back:One page back`

Ask-Function	`elif prog_pos == 'jumping':` `  self.question_bt('Jump',1)`
Visualization	

Submit-Function	`elif prog_pos == 'jumping':`
	`  If button == 'start':`
	`  prog_pos = 'firstpage'`
	`  elif button == 'end':`
	`  prog_pos = 'last_page'`
	`  elif button == 'back':`
	`  prog_pos = settings['Page_History'][-2]`
	`  self.ask()`

`self.question_ls(var,liste,multi=1)`

This function creates a listbox question for variable `var` at position `pos`.

Options:

`var`	String variable denominating the codebook variable to be asked.
`liste`	String variable denominating the codebook variable which contains the list. This may be the same variable as `var` but does not have to be.
`multi`	Integer which may take the values 1 or 0. If set to 1, multiple selection of items is possible.

`self.question_lseek(var,liste,multi=0)`

This function creates a searchable listbox question for variable `var` at position `pos`.

Options:

`var`	String variable denominating the codebook variable to be asked.
`liste`	String variable denominating the codebook variable which contains the list. This may be the same variable as `var` but does not have to be.
`multi`	Integer which may take the values 1 or 0. If set to 1, multiple selection of items is possible.

To search the list the coder may input any string of characters to a textbox above the list. The list will then automatically be reduced to present only the items with this sting inside. The search term is not case-sensitive and does not have to be in full words.

`self.question_ladd(var,liste,multi=0)`

This function creates a searchable listbox question for variable `var` at position `pos` from which single items may be added to an answer list.

Options:

`var`	String variable denominating the codebook variable to be asked.
`liste`	String variable denominating the codebook variable which contains the list. This may be the same variable as `var` but does not have to be.
`multi`	Integer which may take the values 1 or 0. If set to 1, multiple selection of items is possible.

To search the list the coder may input any string of characters to a textbox above the list. The list will then automatically be reduced to present only the items with this sting inside. The search term is not case-sensitive and does not have to be in full words.

To add an item to the selections list or to remove an item, the coder may use the buttons 'Add Item' and 'Remove Item' which are displayed between the two lists.

`self.question_mark_units(var,level)`

This function asks the coder to mark all units of analysis on a given `level`. The exact question is taken from variable `var`.

Options:

`var`	String variable denominating the codebook variable to be asked. This variable has exactly one option which has the label 'Done', 'Finished', or any other confirmation. If there is no such option, it is recommended to leave the Check-button on display.
`level`	String variable denominating the level of analysis on which the units are to be coded.

`self.question_sel_units(var,level)`

This function asks the coder to select a unit of analysis from a given `level`. The exact question is taken from variable `var`. The list of units of analysis is created from text highlights previously stored using the function `self.unit_confirm()`.

Options:

`var`	String variable denominating the codebook variable to be asked.
`level`	String variable denominating the level of analysis on which the units are to be coded.

`self.question_menu(var,position)`

This function creates a menu question for variable `var` at position `pos`. The question takes a hierarchically ordered set of options which may be used similar to the button question. Each option calls the submit function with the parameter `button` set to the code of the selected option.

Options:

`var`	String variable denominating the codebook variable to be asked.
`pos`	Integer variable which may take the values 1, 2, or 3 and which indicates the position of the question on the page.

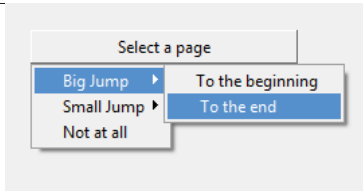
You may use the variable `button` within the Submit-Function to decide upon further actions. The values of this variable correspond to the code of the option selected.

Example:

This short example demonstrates the use of a small menu to jump forward and back in the questionnaire. This function is unlikely to be of any use but it shows the handling of this question type.

Codebook	`[Jump]`
	`Select a page`
	`Jump to another page?`
	`No Helptext available`
	`1:Big Jump`
	`-start:To the beginning`
	`-end: To the end`
	`2:Small Jump`
	`-back:One page back`
	`none:Not at all`

Ask-Function	`elif prog_pos == 'jumping':` `  self.question_menu('Jump',1)`

Visualization	

Submit-Function	`elif prog_pos == 'jumping':`
	`  If button == 'start':`
	`  prog_pos = 'firstpage'`
	`  elif button == 'end':`
	`  prog_pos = 'last_page'`
	`  elif button == 'back':`
	`  prog_pos = settings['Page_History'][-2]`
	`  self.ask()`

7.5. Basic functions

The basic functions are only of interest for developers and should not be tampered with. These functions may be called in some situations; most of them are only called by other functions and are invisible during project development. Nevertheless, the most important ones are briefly introduced here.

`baum_schreiben(direc)`

This function transforms a variable of the type directory to a string variable containing an indented tree representation. `direc` is the name of the directory variable to be transformed.

`bereinigen(uml_string,lc=0,lb=0,uml=0)`

This function takes a string with special characters and transforms it to ASCII. There are some useful options there:

Options:

`uml_string`	String variable or unicode string with special characters.
`lc`	If set to 1, the output is completely in lower case
`lb`	If set to 1, the output contains line-breaks. If set to 0, the line-breaks are replaced by the string `' / '`.
`uml`	If set to 1, the output contains German Umlauts. They may be used in the display of a text but not in the storage thereof. For the purpose of storage, `uml` has to be set to 0.

`self.clean_all_tags(sel_tag=[]):`

This function cleans all highlights from the text display. If desired, only certain types of highlights may be removed by specifying the parameter `sel_tag`.

`self.ausblenden()`

This function removes the text display and all highlight buttons from the GUI.

7.6. Input and output of data

Finally, there is a set of functions which may be used to import and export data in Angrist. These functions may be used to load texts, save data, or import and export tables.

`self.export_data(dta_pos_all, varlist, filename, debug=0)`

This function exports data from the storage directory to rectangular data matrices. The matrices are created as text-files with tabulators as cell delimiters. Each row represents a case. For each level of analysis, an individual may be generated using this function.

Options:

`dta_pos_all`	List of strings containing the level of analysis to be exported and all parent levels of analysis. The order is descending by depth of levels of analysis. Example: If level `'Statement'` should be exported which is below the level of `'Actor'` which is below the level of the text, the value of `dta_pos_all` is: `['Actor','Statement']`. Put differently, the list here represents the odd entries in `dta_pos`.
`varlist`	List of strings denominating all variables to be exported in this table. If any variable name does not exist in the data an empty cell will be exported.
`filename`	A string denominating the file in which the data is to be exported. If the file does not exist yet, a new file will be created with variable names in the top row.
`debug`	If set to 1, all cells will not only contain the codes but also the name of the variable this code belongs to. This may be necessary to debug the export function.

`baum_export()`

This function exports the complete contents of the dictionary `storage` to a file specified in the `settings` dictionary. If there is a valid entry for `settings['Out_Tree']`, an indented representation of the dictionary is written to the file specified. If there is a valid entry for `settings['Out_JSON']`, a JSON-formated representation is written to a file.

`artikelholen(ID)`

This function reads a textfile with name ID to be displayed in the text display. The function always calls the function `textmine()` to which it submits a list of all lines in the text file.

`textmine(linelist)`

This function may be used to set default values of variables on the base of patterns in the text. You may, for example, extract informations from the header of a text, count words, or look for certain actors. This function is called whenever a new text is loaded to the text display.

`get_codebook(filename)`

This function loads a codebook using the format of Angrist codebooks to a dictionary of lists. See chapter 4.3 for detailed information.

`get_todo(filename)`

This function reads a list of text IDs to return the topmost entry. This function is used to get the next item on the todo list.

`check_todo(filename)`

This function changes a todo list in a given file by removing the current text ID from the list. This function may be called at the end of data collection to update the todo list.

get_data(filename, varlist=[])

This function reads a data matrix stored in a text file with tabulators as cell delimiters. The data is stored in a dictionary of lists which each contain all cells in a column. The key for each list is the topmost entry (usually the name of the column).

Options:

`varlist` List of variables to be loaded. If omitted, the first entry is used as column name.

get_varnames(filename)

This function extracts the header of a data matrix stored in a text file with tabulators as cell delimiters. This header usually contains the names of the columns.

write_data(data, varlist, filename)

This function writes a data dictionary (as produced by `get_data()`) to a text-file with tabulators as cell delimiters.

8. Impressum

Programmer:

Martin Wettstein

Institut für Publizistikwissenschaften und

Medienforschung der Universität Zürich (IPMZ)

Andreasstrasse 15

8050 Zürich

m.wettstein@ipmz.uzh.ch

Tel: 044 635 20 78

Anhang B2: Nogrod

Online-Ressource:

Im Folgenden wird die Version bei Einreichung dieser Dissertation dargestellt. Eine vollständige und ständig aktualisierte Version der Dokumentation und Anleitung, inklusive Beispielprogramme, sind Online verfügbar unter:

http://www.ipmz.uzh.ch/de/Abteilungen/Medienpsychologie/Recource

Nogrod 0.3 (beta)

Quick tutorial

Martin Wettstein

Nogrod, n [nogrod] 1. Python script to reshape and analyze datasets in text format prior to importing them to statistical software packages. 2. (from Sindarin: *hollow-bold*): Ancient dwarf city in the Blue Mountains, renowned for skilled craftsmen. Popular exports of the smiths in Nogrod were the Nauglamir and Angrist.

Zürich, June 2015

Contents

Chapter 1. Introduction **3**
1.1. Example Data 3
Chapter 2. Handling of Nogrod **5**
2.1. Help texts 6
2.2. List selection 6
2.3. Starting Nogrod (only for Division Media Psychology & Effects) 7
2.4. File handling 7
2.5. Header and Separator 8
Chapter 3. Merging Procedures **9**
3.1. Add Cases 9
3.2. Add Variables 10
Chapter 4. Subset Procedures **11**
4.1. Subset of cases 11
4.2. Random Subset of cases 11
4.3. Subset of Variables 12
Chapter 5. Reshaping Procedures **13**
5.1. Dummy Table 13
5.2. Reshape to slim table 14
Chapter 6. Aggregation Procedures **16**
6.1. Aggregate Cases 16
6.2. Aggregate Variables (Calculation) 17
6.3. Calculate Entropy within groups 19
Chapter 7. Transformation Procedures **22**
7.1. Transform Timestamps 22
7.2. Transform Scale to Groups 23
Chapter 8. Co-Occurrence Analysis **26**
8.1. Co-Occurrence analysis from multinomial variables 26
8.2. Co-Occurrence analysis from Dummy-tables 27
Chapter 9. Social Network Visualization **29**
9.1. Social Network Visualization from two variables 29
9.2. Visone for Nino Abzianidze 30
9.3. Handling Visone 30
Chapter 10. Time Series Analysis **37**
10.1. Detect Peaks 37
10.2. Flatten moving Average 38
10.3. Create Gliding Window 39
Chapter 11. Sequence Analysis **40**
11.1. Find common sequences 40
11.2. T-Pattern Analysis 41
Chapter 12. Cluster Analysis **42**
12.1. Cluster Analysis of Count data (HECATE) 42
Chapter 13. Reliability Testing **44**
13.1. Test interrater reliability 44

Chapter 14. Examples **49**
14.1. Add cases from another dataset 49
14.2. Add information on the actors to the table 50
14.3. Select statements of democrats containing an attack 51
14.4. How many times was each actor cited on each station? 51
14.5. Which issues were covered in each of the TV shows? 52
14.6. Which were the most frequent actors and issues on each station? 53
14.7. Reshape the dataset to one show per line 53
14.8. Which arguments were seen on which show? 54
14.9. How many arguments are there in the statement? 55
14.10. Diversity of the networks with respect to actor affiliation 56
14.11. Actor Diversity across networks 56
14.12. How probable is the discussion of certain different issues on the same show? 57
14.13. Which arguments are often mentioned together? 57
14.14. Finding peak skin conductance during a movie 59
14.15. Find peak increases in skin conductance 60

Chapter 1. Introduction

Nogrod is a script for Python 2.7 which uses the tk-interface of the current platform to enable quick and easy data processing and data manipulation. The script was designed to process and reshape data stored in textfiles (CSV or tabstopp separated) without the need to open them in statistical software packages. The program has a few easy to use routines which allow aggregation, combination, cross tabulation, and some basic analysis functions such as contingency analysis, for numerical and non-numerical data.

The program is based on the Angrist engine for content analysis data input and is specifically designed to read and process the data stored from Angrist. It may, however, be used for any kind of data which is stored in textfiles.

1.1. Example Data

In this quick tutorial a small and wholly artificial dataset is used to explain and demonstrate the basic functions of Nogrod. The data is designed to look a little like content analysis data on three US TV-networks and their quotation of political actors on three issues. All data (except for the actor affiliations) is fictional and solely for education purposes.

The whole example dataset may be found in the file 'Example_Data.txt'.

Table 1: Example Data for this quick reference. The table contains 14 variables and 19 cases as may be obtained by quantitative content analysis of evening news. The data is purely fictional and just meant for illustration purposes.

Show_ID	Station	Date	Actor	Actor_Affiliation	Quotation	Issue	Argument_Moral	Argument_Economy	Argument_Voter	Argument_Justice	Argument_Faith	Attack	Attack_Target
FN001	Fox	Mar 13	Obama, Barrack	Dem	Direct	Budget	1	1	0	0	0	0	
FN001	Fox	Mar 13	Van Hollen, Chris	Dem	Indirect	Budget	0	0	0	0	0	0	
FN001	Fox	Mar 13	Ryan, Paul	Rep	Direct	Budget	0	0	0	1	0	1	Obama, Barrack
FN001	Fox	Mar 13	Cruz, Ted	Rep	Direct	Obamacare	0	1	0	0	0	1	Obama, Barrack
FN002	Fox	Mar 14	Christie, Chris	Rep	Indirect	Obamacare	1	0	1	0	0	0	
FN002	Fox	Mar 14	Ryan, Paul	Rep	Direct	Budget	0	0	0	0	1	1	Biden, Joe
FN002	Fox	Mar 14	Kerry, John	Dem	Indirect	Budget	1	0	1	0	0	0	
FN003	Fox	Mar 20	Christie, Chris	Rep	Direct	Iran	0	0	0	0	1	1	Foreign Actor
PB001	PBS	Mar 13	Van Hollen, Chris	Dem	Direct	Budget	0	1	0	1	0	0	
PB002	PBS	Mar 15	Obama, Barrack	Dem	Indirect	Iran	0	0	0	0	0	1	Foreign Actor
PB002	PBS	Mar 15	Cruz, Ted	Rep	Direct	Iran	0	0	0	0	0	0	
PB003	PBS	Mar 20	Van Hollen, Chris	Dem	Direct	Budget	0	1	0	1	0	0	
PB003	PBS	Mar 20	Obama, Barrack	Dem	Indirect	Obamacare	0	0	0	0	0	0	
AB001	ABC	Mar 14	Ryan, Paul	Rep	Direct	Obamacare	0	0	0	0	1	0	
AB001	ABC	Mar 14	Kerry, John	Dem	Direct	Obamacare	0	0	0	1	0	0	
AB001	ABC	Mar 14	Cruz, Ted	Rep	Direct	Obamacare	0	0	0	0	1	0	
AB001	ABC	Mar 14	Van Hollen, Chris	Dem	Direct	Budget	0	1	0	0	0	1	Tea Party
AB002	ABC	Mar 15	Obama, Barrack	Dem	Direct	Iran	1	0	0	0	0	0	
AB002	ABC	Mar 15	Kerry, John	Dem	Indirect	Iran	0	0	1	0	0	1	Foreign Actor

In addition to this example data, there is a second table holding additional information on certain actors, which is used as an example for the merging function (add variables). This table is considerably small and just contains the functions of the actors (Table 2)

Table 2: Example Functions. This table contains additional information on the actors in Table 1. The data is actually accurate but by no means systematic. It is just meant for demonstration purposes.

Actor	Function
Obama, Barrack	President
Van Hollen, Chris	Representative
Ryan, Paul	Representative
Cruz, Ted	Senator
Christie, Chris	Governor
Kerry, John	Secretary of State

Since this table contains only few cases and just four different timestamps (March 13, 14, 15, and 20), this data would not be useful in time series analyses. For this reason, there is another example dataset which contains time series data. The table has three variables (see Table 3) and 2319 cases.

Table 3: Variables in time series example data

Variable	Description	Range
Time	Timestamps in seconds	30.12-319.88 Rate=8Hz
SkinCond	Skin Conductance in micro-Siemens for each measruement point	1.97-2.65 M=2.114; SD=.136
Heartrate	Heartrate in beats per minute, calculated once per second	55.45-80.00 M=67.44; SD=4.11

Since the whole table may not be shown here for space issues, the data is visualized in two line graphs for the two time series variables (see Figure 1).

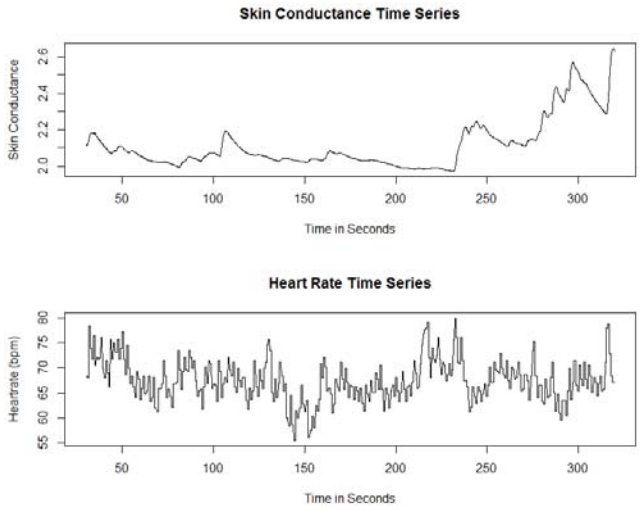


Figure 1: Time Series Data visualization.

Chapter 2. Handling of Nogrod

Nogrod was designed to offer an intuitive interface for data processing. Since each kind of data processing requires a set of parameters (such as input data, procedure, method, output format...), the user is required to provide some input on the data processing routine to be performed. As in Angrist, the user is asked to enter each of the parameters in a sequence of easy to answer questions with help texts. After each confirmation, Nogrod checks the validity of the parameters and provides the user with additional information on the data and potential risks when using a parameter which does not fit the data entered.

The screen of Nogrod always has the same structure (See Figure 2) to enhance usability and to keep it as simple and straightforward as possible.

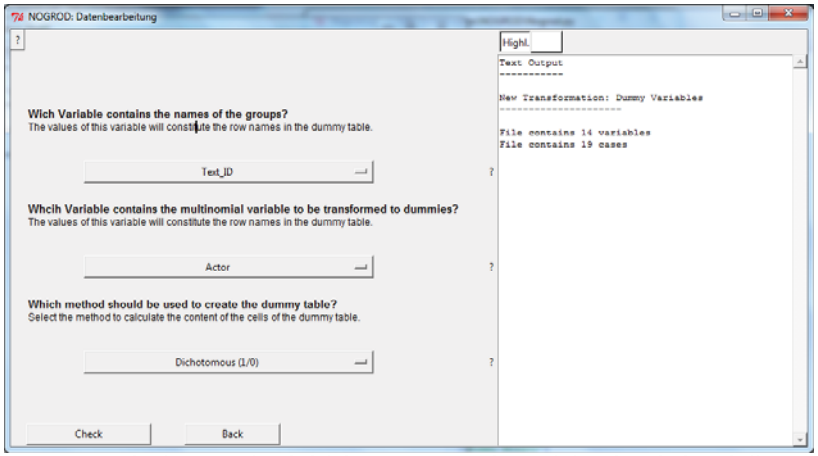


Figure 2: Screenshot from Nogrod on the left the parameters may be specified, on the right the tool offers additional information on the data and current processes.

In figure 1 you see the screen as it presents itself during the parameter entry for a dummy transformation. On the right hand side there is a console providing text output on the current data processing routine. In this example, the user has loaded a dataset with 14 variables and 19 cases.

On the left hand side, the program asks for three inputs to be made for this data processing method. First, it asks for the two variables to be included in the dummy transformation. The variables may be selected from a dropdown menu containing all variables in the current dataset. Second, the program asks for the exact method of dummy transformation to be used. The method may be chosen using the dropdown-options provided.

Upon entering all data, you may confirm using the button 'Check' at the bottom of the left-hand side of the program. If a previous selection was wrong, you may also choose to go back by one page using the button 'Back'.

Attention: Upon confirming the entries with 'Check', the program performs a set of preliminary analyses on the data to ensure the validity of all parameters. For large datasets this may take some seconds in which the check button appears sunken and the program does not react to any

input. Please be patient and wait for the program to finish its task. After the last page, which usually includes the specification of an output file, the program performs the whole transformation. For large datasets this may take several minutes. Please be patient and don't force the program to close.

Upon completion of a data processing routine, the program asks whether you want to proceed with another routine. You may continue using the data you just created or use another dataset for analysis or transformation.

2.1. Help texts

At the right-hand side of each question, a small question mark is visible. These question marks provide additional information on the parameter to be entered. Just click them to open a pop-up window with additional information (see Figure 3).

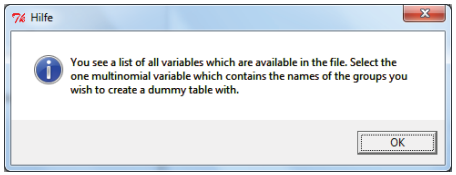


Figure 3: Help pop-up for the Group Variable

2.2. List selection

In most routines some parameters are entered as lists of objects you wish to include in the analysis. These lists may consist of variables, values, or cases. To enter these lists as quickly as possible you are presented with a list containing all possible selections and an empty list of selected items (see Figure 4).

You may add single items to the list at the bottom by selecting them and pressing 'Add Item'. The item will then be added to the bottom list but not removed from the list of possible selections. If you want to add all items to the selection you may press 'Add all'. If an item is added to the selection which you don't want to add, you may press 'Remove Item' to remove it from the selection again. This enables you to select all but one item by adding all items first and then removing the one item you wish to exclude. By pressing 'Remove All', the selection list will be cleaned and you may start selecting again.

Please note that for some list selections it is necessary to include at least one element to the selection list. If no item is added, the programm will not accept the entry and prompt you to select an item.

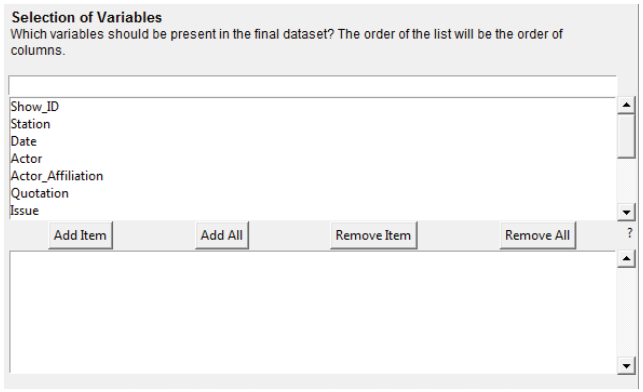


Figure 4: Screenshot of the list selection. In the top list all possible selections are presented. You may add and remove them to the selection list at the bottom using the four buttons between the lists

2.3. Starting Nogrod (only for Division Media Psychology & Effects)

Double-click the file:

G:\[IPMZ-intern]\Abteilung V\[Interner Bereich]\Forschung\Python_Programme\Nogrod.bat

2.4. File handling

For loading and storing files, Nogrod allows for two different ways. First, you may enter a filename directly to the textbox. If you want to enter the whole path, please use slashes (/) instead of backslashes (\) to separate directories. This applies to windows as well, which usually does not support slashes. Backslashes may not be recognized properly.

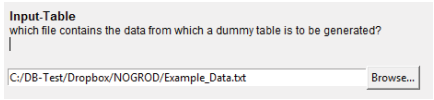


Figure 5: Input of filenames

If you don't know the exact filename you may click 'Browse...' to look up the file on your computer. Depending on the platform and version, the dialog box which opens may differ. It is usually identical to the dialogs used in programs on this computer (e.g. MS Word).

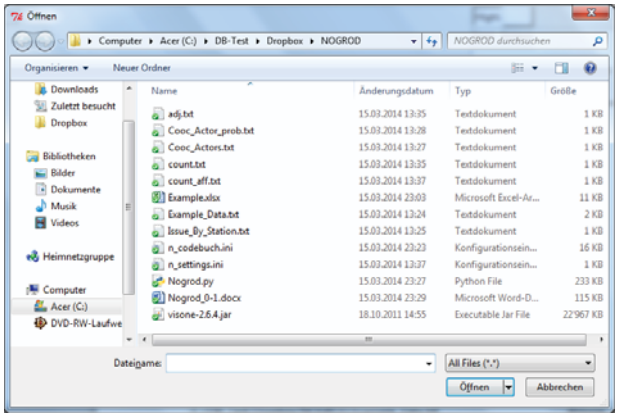


Figure 6: File dialog. This dialog may differ depending on the platform Nogrod is used on.

2.5. Header and Separator

When tables are stored in text files, they may have first line (a header) containing the name of each variable. This line should contain the same number of entries as each row below it and each name should only occur once. If any name occurs twice in a file, Nogrod will automatically rename the variables to make them unique by adding a number to each duplicate. If no header is present, each variable will be assigned a number.

Depending on the style of the text file, the columns (or cells within a row) are separated by a special character. In output files from Angrist, the separator is always a tabstopp. The same applies to text files which are created by copy&pasting values from an Excel table. In CSV files, the cell delimiter or separator is a semicolon (;). In BASIC tables, the separator is a comma (,) and sometimes a blank space is used to separate columns.

When opening or storing any file in Nogrod you are asked to specify whether this file has a header and which separator is used. Default values are 'Yes' and 'Tabstopp'.

Chapter 3. Merging Procedures

Merging procedures may be used to combine data from two different tables, both stored in textfiles, to one single table. Nogrod allows for two different ways of merging tables. First, you may add cases (rows) from another table to a working table which contains the same variables. Second, you may add variables (columns) from a key table to the working table by combining the cases over a common key variable.

3.1. Add Cases

If you want to combine two tables which contain (largely) the same variables you may use the merge procedure 'Add Cases'. You may open two tables simultaneously, select the variables which should be present in the resulting table and combine the two tables.

Note that you may combine tables with different variables. Some of them should, however, be present in both tables. For cases which come from a table which does not contain all variables of the output table, the values of the missing variables will be set to missing (blank cells).

Applications:

This procedure may be used whenever you want to:

- Combine fragments of the same dataset which were separated
- Combine two different datasets with the same variables
- Add some cases to an existing dataset which were stored somewhere else
- See Example: *Add cases from another dataset*

Options:

First Table

This table is a text document containing the first part of the table you wish to merge. In the end, the cases in this table will be at the top of the table, while the cases of the second table are at the end.

Second Table

This table is a text document containing the second part of the table you wish to merge. The second table should contain some variables which are present in the first table as well. If this is not the case, the script will work properly but the resulting table is hardly usable for further analysis as there will be many missing values.

Variables in the resulting table

From a list of variables which were present in either table, you may select any number of variables. Note that a variable which only occurs in one table will have missing values for all cases from the other table.

Order of the variables: In the resulting table, the order of the variables will be according to the order in the source tables. First, there are all variables which were present in the first table, then there will be all variables which were present in the second table only.

Hint: If you only want to add variables which are present in both tables, enter 'both' at the textbox above the list. The list will then be reduced to items which are present in both tables (unless you have a variable which contains the string 'both'. These will be shown as well. You may then press 'add all' to add all variables in the reduced list to the selection.

Output File

The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

3.2. Add Variables

Another way to merge tables is the addition of variables to an existing table by combining the cases using one or several key variables. By this procedure information from a key table is inserted to a new variable of a main table. This may be used to add context information on recurring values of a variable in the key table or to add seperate measures for all cases of the main table which were stored in another table.

Applications:

This procedure is not suited to answer questions but it is really useful to reshape and combine existing datasets. The most common applications are:

- Combine two tables containing the same cases
- Combine two tables containing mutual variables but different cases
- Add additional information on specific values of a multinomial variable to a dataset
- Recode a variable according to special rules
- Attach dummy-variables produced with the above procedure to the source file (transform a multinomial variable to a set of dummy variables)
- See Example: *Add information on the actors to the table*

Options:

Main Table

This table contains the main dataset to which you wish to add information from a key table. At least one variable of this table has to contain values identical to one variable in the key table to perform matching. It may also contain several variables which may be used as key variables. The table does not have to be sorted.

Key Table

This table contains additional information which may be added to cases in the main table by means of a key variable. The key table may contain any number of variables of which you may add some or all to the main table. Neither this table nor the main table needs to be sorted.

Key variables in main table

You may select any number of key variables which are used to identify a case. Only cases with identical values on all key variables are matched. Cases for which no corresponding case exists in the key table are assigned missing values for all added variables.

Key variables in key table

As with the key variables in the main table you may select multiple variables. You must select the corresponding variables in the same order as the key variables in the main table. The key variables may be named differently in both tables.

If there are duplicates (i.e. cases with identical values on all key variables) in the key table, you will be warned. Only the last of the duplicates is matched with cases in the main table.

Variables to be added

When you have selected the key variables in both tables you may select any number of variables in the key table to be added to the main table. You may only add variables which are not among the key variables (as they are present in the main table anyway).

Output File

The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

Chapter 4. Subset Procedures

Subset procedures may be used to create a smaller table from an existing table, which does not contain all cases or variables. This may be useful for large tables or for tables with different categories of entries which should be separated prior to analysis.

4.1. Subset of cases

The first subset procedure allows you to select cases from a table, based on their values on certain variables. You may select any number of variables and any combination of values which should be present in the subset to be extracted. The dataset does not have to be sorted in any way prior to subset extraction.

Applications:

Especially with large datasets it may be required to select only some cases for further analysis. These cases may be defined by one or several variables. For the example dataset in this tutorial, automated subset extraction does not make much sense as there are only 19 cases to begin with, but for the sake of testing the procedure you may use it to:

- Select only shows which were issued on Fox News
- Select statements which contain an attack
- Select statements which were made by a democrat on March 14
- See Example: *Select statements of democrats containing an attack*

Options:

Input Table

This is the table which should be reduced to a subset. It should contain one or several multinomial variables defining groups of cases which may be extracted.

Grouping Variables

The grouping variables define the group which is to be extracted. You may select any number of variables wich are necessary to define the subset you wish to extract. The order of grouping variables is irrelevant for this method.

Grouping Values

From a list which contains all combinations of values of the selected grouping variables you may select the combinations which belong to the subset you wish to extract. You may select any number of combinations. Use the textbox above the list to filter the combinations.

Output File

The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

4.2. Random Subset of cases

The second function draws a random sample of cases from a list. You may select any size of the sample. The output table is not sorted in any way but contains the selected cases in a random sequence.

The randomized output of cases results in a sometimes handy side-effect of this function. You may randomize the order of cases in your dataset by drawing a random sample exactly the size of your dataset.

Applications:

In order to test and perform analyses which are very time-consuming, it may be required to divide large datasets to smaller random samples. This is especially the case for language processing or other expensive routines.

Options:

Input Table
This is the table which should be reduced to a subset. It should contain one or several multinomial variables defining groups of cases which may be extracted.

Number of Cases
You may enter any number of cases to be drawn from the table. If you enter a number which is higher than the total number of cases in the table, the only effect of this function will be to randomize the cases in your table.

Output File
The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

4.3. Subset of Variables

Another way for creating a subset from a table is the selection of certain variables. This function may prive useful for large datasets where only a small part of the variables is required in an investigation.

The usage of this procedure is quite straightforward, as you only open a file and select a number of variables to retain. Therefore, examples are omitted, here.

Options

Input Table
This is the table which should be reduced to a subset. It should contain several variables, some of which you wish to extract.

Selection of Variables
From a list containing all variables in the dataset you may select the variables you wish to extract to retain in the subset.

Output File
The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

Chapter 5. Reshaping Procedures

Reshaping procedures may be used to transform the shape of tables. The main use of these reshaping procedures is the transformation of a multinomial Variable to a series of dichotomous variables (Dummy Table) or the transformation of a dummy table to one multinomial variable providing the same information (Slim Table). By either transformation, all information is retained but visualized in a slightly different way.

5.1. Dummy Table

The aggregation of cases containing a multinomial variable may be done using the dummy table routine. Dummy tables are large contingency tables of two multinomial variables where one variable denominates the groups (rows) and the other one the elements occuring within groups (columns) the cells provide information on whether the element occurred in a specific group (dichotomous) or how many times it occurred in a specific group (count data).

The result of a dummy transformation is a new dataset which contains one case per group and one variable for each value of the multinomial variable defining the elements. This dataset may then be used for further data processing or analysis.

Applications:

Dummy tables provide answers to a specific kind of questions: "(How many times / Where) does this value occur in these groups?"

With regards to the example dataset you might answer questions such as:

- On which issues does each speaker make statements?
- On which dates were the different issues covered?
- See Example:
- *Which issues were covered in each of the TV* shows?
- See Example: *How many times was each actor cited on each station?*

Options:

Input Table
This is a table containing at least two multinomial variables. One signifies the groups which are to be considered, the other one holds N different values which should be transformed to N dichotomous variables.

Grouping Variable
This variable contains the names of the groups within which the values of the multinomial variable should be found or counted. In extreme cases, the grouping variable may contain exactly one value which will result in a dummy table containing only one row or it may have a different value for each case which will result in a dummy table the same length as the input table.

The former case may be used to create a frequency table for a variable. The latter case may be used to simply transform a variable to a series of dummies.

Multinomial Variable
The multinomial variable may take at least two values. The values of this variable will constitute the columns of the resulting dummy table.

Method for cell contents

You may choose different methods to calculate the values of the cells of the resulting dummy variable.

- **Dichotomous**:
  The cells of the resulting table may contain 1 (the value is present in the group) and 0 (the value is not present in the group.
- **Count**:
  The cells of the resulting table contain the exact number of times a value is present within each group.
- **Logarithmic**:
  The cells of the resulting table contain the logarithm of the number of times a value is present within each group. This method is especially useful for large counts. In order to prevent log(0)-errors, the count data is increased by 1 to calculate the logarithm. Thus, the value for 0 occurrences is log(1)=0.

Minimal number of elements per group

You may select the minimal number of elements each group has to contain. Groups containing less elements are removed from the dummy table. This means that the sum of values in each row is at least as high as the number you select here.

Note that the sum of a row containing dichotomous values is naturally lower than the sum of a row containing count data. Depending on the method you may have to adjust the minimal number of elements.

Minimal number of occurrences per value

You may select the minimal number of occurrences for each value of the multinomial variable to justify a dummy variable. Values occurring less than the specified number are removed from the table. This means that the sum of each column is at least as high as this number.

As the minimal number of elements per groups and the minimal number of value occurrences have to be fulfilled at the same time, sparse rows and columns are removed until both requirements are met.

Note that the sum of a column containing dichotomous values is naturally lower than the sum of a column containing count data. Depending on the method you may have to adjust the minimal number of elements.

Dummy Output

The resulting dummy table will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

5.2. Reshape to slim table

The inverse function of a dummy transformation is the transformation of a series of dichotomous or count variables to one single multinomial variable. There are essentially two ways of reshaping a dummy table to a slim table. First, you may choose to interpret the values of the dummy table as counts and reproduce a table in which the multiniomial variable appears the respective number of times in each case. Zeroes in the dummy table will result in no information in the slim table. Second, you may choose to interpret the values of the dummy table as values to be transferred to the slim table as is. In that case, two variables are created in the slim table, one holding the name of the original dummy variable, the other holding its value.

For illustration purposes, take the first few cases of the Example_Data as a dummy table to be transformed. By regarding the values of

Table: Transformation example: The dummy table may (left) may be transformed by interpreting the values as counts (middle) or values (right). The table on the right is incomplete for formatting reasons as it would have 25 lines.

Actor	Argument_Moral	Argument_Economy	Argument_Voter	Argument_Justice	Argument_Faith
Obama, Barrack	1	1	0	0	0
Van Hollen, Chris	0	0	0	0	0
Ryan, Paul	0	0	0	1	0
Cruz, Ted	0	1	0	0	0
Christie, Chris	1	0	1	0	0

Actor	Argument
Obama, Barrack	Moral
Obama, Barrack	Econom
Ryan, Paul	Justice
Cruz, Ted	Econom
Christie, Chris	Moral
Christie, Chris	Voter

Actor	Argument	Value
Obama, Barrack	Moral	1
Obama, Barrack	Econom	1
Obama, Barrack	Voter	0
Obama, Barrack	Justice	0
Obama, Barrack	Faith	0
Van Hollen, C.	Moral	0
Van Hollen, C.	Econom	0
Van Hollen, C.	Voter	0
...	...	...

Options:

Input Table

This is a table containing at least two dummy variables with dichotomous or count data. Additional variables are of no consequence and will be copied to the output table if not used in the reshaping process.

Variables to be condensed

From all variables in the dataset you may select the ones which should be condensed to one multinomial variable.

Type of Reshaping

Select whether the values should be interpreted as count data or values to be copied to the output table. In the latter case, two variables will be created.

Output File

The resulting dummy table will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

Chapter 6. Aggregation Procedures

Aggregation procedures allow you to decrease the size of tables by aggregating cases or variables according to fixed rules. For all aggregation procedures aggregating cases you need a group variable which indicates the groups to which the cases are to be collapsed. You may then select the variables to be aggregated and the function to be used.

For the aggregation of values you may select a series of variables which are to be combined to one outcome variable. A small set of functions (e.g. mean score, sum, concatenation) may be used to aggregate the variables.

6.1. Aggregate Cases

The aggregation of cases to groups may be used when several variables have to be aggregated using a common rule (e.g. Sum of values). The cases will be combined to groups specified by the group variable and will contain one value for each variable selected for aggregation. This value is computed from the single values of all cases within the group by a rule you may select.

This data transformation is especially useful when handling relational datasets from content analyses where you may have several cases in one table which have to be aggregated to combine them with another table containing group specifics. In the example dataset you have several actors per news story. If you wish to combine this data to another table containing additional information on the news stories, you may aggregate the dataset first with the news story as a group and then merge the resulting dataset with another table.

You may also use this procedure to get mean scores or sums of values for certain groups without using statistical software. You may also use this procedure to reshape a dataset by copying multiple units of analysis next to each other on the same line in the dataset.

Applications

- See Example: *Which were the most frequent actors and issues on each station?*
- See Example: *Reshape the dataset to one show per line*

Options:

Input Table

This file contains a table with at least two variables. One variable defines groups and the other contains values which may be summed up or otherwise aggregated within these groups.

Grouping Variables

From a list containing all variables in the dataset you may select any number of variables defining the groups you wish to aggregate the dataset on. If you select more than one variable, each unique combination of values of these variables defines one group.

Variables to be aggregated

From a list containing all variables except for the grouping variables, you may select any number of variables which should be aggregated for each group.

Method for aggregation

Depending on the type of variable which is to be aggregated, you may select one of the following methods to calculate one value per variable per group. Please note that Sum, Mean, Maximum and Minimum are possible only for numerical data. The result of these methods is always a floating number.

- **Sum**: For each variable selected, the sum will be calculated for each group. Missing values are ignored.
- **Mean**: For each variable selected, the mean value will be calculated for each group. Missing values are ignored (i.e: mean(1,2, MISSING ,3) =2)
- **Maximum**: For each variable selected, the highest value will be selected for each group.
- **Minimim**: For each variable selected, the lowest value will be selected for each group.
- **Most Frequent**: For each variable selected, the most frequent value will be selected for each group. If there are two most frequent values, the choice will be the value which is lower (either numerically or in the alphabet)
- **First Value**: For each variable selected, the first value in the group within the dataset is selected.
- **Last Value**: For each variable selected, the first value in the group within the dataset is selected.
- **Transform Dataset to broad format**: Each variable selected will be multiplied to copy all values completely to one line in the new dataset. The number each variable is copied depends on the highest number of cases per group in the input dataset. When aggregating the example data in this way to get one show per line, each variable would be copied four times. See example

Note that you may only select one method for all variables selected. If you want to use one method for a set of variables and another method for another set, please do this separately and use the merge routine to combine the resulting datasets.

Output File

The resulting table will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

6.2. Aggregate Variables (Calculation)

The aggregation of variables is a quick way to generate new variables from existing ones using five standard methods (sum, mean, min, max, and concatenate). You may use this procedure to create new variables without using statistical software packages or copy&pasting the data to Excel.

The range of the easy calculation proceture is deliberately limited to the five basic methods to keep it simple. For more sophisticated methods, you will have to use calculation programs (such as Excel) or statistical software.

Applications:

Easy calculation may be useful in cases where you need the sum or mean value of a certain variable for additional analyses. In these cases it may save time to perform the calculations directly in Nogrod instead of importing and exporting the data to an other program.

In the example data there is only a small number of numerical variables to use:

- Was there any arugment present?
- How many arguments are there in the statement?
- See Example: *How many arguments are there in the statement?*
- See Example: *Reshape* the dataset to one show per line
- *Here, the data is summarized to* the level of shows and all variables are aggregated by reshaping them to a broad view. For demonstration purposes and graphic limits, only three of the variables (Actor, Actor Affiliation, and Issue) are used.
  1. Select 'Aggregate Data' -> 'Aggregate Cases' as a procedure and load the example data.
  2. Select 'Show_ID' as the group variable.
  3. Add Actor, Actor_Affiliation, and Issue as variables to be aggregated. Select 'Transform Dataset to broad format' as method.

4. Choose an output file and confirm.

- The result is quite a broad table as all variables have been multiplied four times to make space for all data which was aggregated. The resulting table contains four copies of each variable as there is a maximum of four speakers in the shows of the example data. The aggregated lines are added in a chronological order, so Actor01 of AB001 is Ryan, followed by Kerry, Cruz, and Van Hollen. For Groups with less than four cases, missing values are entered to unused variables. As FN003 only has one actor, this case has nine missing values which might be used to specify actor 02, 03, and 04.

Reshape the dataset to one show per line
Here, the data is summarized to the level of shows and all variables are aggregated by reshaping them to a broad view. For demonstration purposes and graphic limits, only three of the variables (Actor, Actor Affiliation, and Issue) are used.
Select 'Aggregate Data' -> 'Aggregate Cases' as a procedure and load the example data.

5. Select 'Show_ID' as the group variable.
6. Add Actor, Actor_Affiliation, and Issue as variables to be aggregated. Select 'Transform Dataset to broad format' as method.
7. Choose an output file and confirm.

The result is quite a broad table as all variables have been multiplied four times to make space for all data which was aggregated. The resulting table contains four copies of each variable as there is a maximum of four speakers in the shows of the example data. The aggregated lines are added in a chronological order, so Actor01 of AB001 is Ryan, followed by Kerry, Cruz, and Van Hollen. For Groups with less than four cases, missing values are entered to unused variables. As FN003 only has one actor, this case has nine missing values which might be used to specify actor 02, 03, and 04.

Table 12: Result of the aggregation to broad format. The table is cut in two parts as it is too broad to be displayed on one line.

Show_ID	Actor01	Actor_ Affiliation01	Issue01	Actor02	Actor_ Affiliation02	Issue02
AB001	Ryan, Paul	Rep	Obamacare	Kerry, John	Dem	Obamacare
AB002	Obama, Barrack	Dem	Iran	Kerry, John	Dem	Iran
FN001	Obama, Barrack	Dem	Budget	Van Hollen, Chris	Dem	Budget
FN002	Christie, Chris	Rep	Obamacare	Ryan, Paul	Rep	Budget
FN003	Christie, Chris	Rep	Iran			
PB001	Van Hollen, Chris	Dem	Budget			
PB002	Obama, Barrack	Dem	Iran	Cruz, Ted	Rep	Iran
PB003	Van Hollen, Chris	Dem	Budget	Obama, Barrack	Dem	Obamacare

Actor03	Actor_ Affiliation03	Issue03	Actor04	Actor_ Affiliation04	Issue04
Cruz, Ted	Rep	Obamacare	Van Hollen, Chris	Dem	Budget

Ryan, Paul	Rep	Budget	Cruz, Ted	Rep	Obamacare
Kerry, John	Dem	Budget			

Which arguments were seen on which show?

Options:

Input Table
This file contains a table with more than one variable.

Variables to be aggregated
From a list of all variables in the dataset you may select any number of variables you wish to aggregate to one variable.

Method for calculation
Depending on the type of the variables you wish to aggregate you may select different methods for aggregation.

- **Sum of all values**: Calculates the sum of numerical variables
- **Mean of all values**: Calculates the mean of numerical variables
- **Maximum value**: Uses the highest value of the numerical variables
- **Minimum value**: Uses the lowest value of the numerical variables
- **Concatenate**: Paste the values together in one large string

The contatenation is the only method which may be used on string variables. If one of the variables contains string values in one of the other methods, these values are treated as missing values. In the calculation of sums, missing values are treated as 0. In the calculation of means, missing values are ignored, not adding to the number of values to calculate the mean of.

Name of output variable
The aggregation of variables results in an additional variable containing the results of the calculation. You may enter a name for this variable. It will be put at the end of the table.

Output File
The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file. The table is identical (complete and in order) to the input table except for the additional variable at the end of the dataset.

6.3. Calculate Entropy within groups

A special way to aggregate the cases in your data for one multinomial variable is the calculation of the group entropy, meaning the diversity of values on a given variable within a group. Calculating the group entropy will result in one value for the variable selected.

This method may be used to determine the diversity of groups in your data, which may be useful for content analyses addressing the diversity (or focusedness) of media content.

As with dummy tables you may specify one variable which defines the groups and one multinomial variable which is used to determine the entropy.

Calculation:

The entropy of a group is calculated using this formula, where N is the number of unique elements in the group and M is the number of unique elements in the population. p_i is the proportion of element i in the group.

$$Entropy = -\sum_{i}^{N} p_i * log_M(p_i)$$

If we assume two different elements within a group where the first appears twice (p_1=0.66) and the second appears once (p_2=0.33), the entropy would be:

$$\boldsymbol{Entopy} = 0.66 * log_2(0.66) + 0.33 * log_2(0.33) = \mathbf{0.919}$$

The Entropy may assume any value between 0 (perfect uniformity) and 1 (perfect diversity).

Applications:

- See Example: *Actor Diversity across networks*
- See Example: *Diversity of the networks with respect to actor affiliation*

Options:

Input Table

This file contains a table with a grouping variable and a multinomial variable for which the entropy within each group is to be calculated.

Grouping Variable

From a list containing all variables in the dataset you may select the variable defining the groups. In order to be considered a group for the calculation of entropy, at least two cases have to belong to it. Groups consisting of only one element are assigned a missing value for entropy.

Multinomial Variable

From a list containing all variables but the grouping variable you may select the variable for which you want to calculate the entropy of values within each group. This variable should have at least two different values and there should be cases with the same value on this variable to result in useful output.

List of Reference

When calculating the entropy within a group, there are two different ways for this calculation. On one hand, you may calculate the entropy of the elements present in this group without considering other groups. On the other hand, you may see the elements within a group as a selection from a more extensive universe of elements. This selection itself decreases the entropy.

- **All elements across all groups**: This option calculates the entropy for each group using all possible values of the variable as population. This means that the population of possible values is the same for all groups and the entropy values are typically lower as no group is likely to contain all values.
- **Only the elements within each group (M==N)**: This option calculates the entropy for each group based on the elements which are present in this group. A group containing two different values has entropy of 1.0 as the values are equally distributed. Even if there are ten different values in another group.

Output file

The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

Chapter 7. Transformation Procedures

Transformation procedures may be used to recode single variables accoriding to simple patterns. In this version of Nogrod, there are two different transformation procedures which may prove useful. First, there is a function to transform timestamps from one standardized format to another. Second, there is a transformation function which allows the grouping of scalar variables.

7.1. Transform Timestamps

The transformation of timestamps may be used to change the format of timestamps stored in a table to another format. For example, you may use them to recode German dates to American dates.

The function is straightaway and only asks you for an input variable which contains timestamps and an output variable in which the new timestamps should be written. Then you may select the input and output format of the timestamps.

Options:

Input Table
This file contains a table with one variable containing timestamps in any format which is to be converted to another format.

Input Variable
From a list of all variables in the dataset you may select the variable containing timestamps.

Output Variable
As a new variable is created by the transformation, you may specify the name of the new variable by entering it manually.

Format of input variable
There is a set of standardized timestamps from which you may select the timestamp format of the input variable.

- **Python Timestamp (string)**: This format is created by the time.ctime() function in python and has the form "Weekday Month Day Time Year" (e.g.: "Fri Mar 14 10:41:57 2014"). This timestamp may be found in Angrist outputs.
- **Python Timestamp (seconds)**: This format is created by the time.time() function in python and indicates the seconds which have passed since January 1 in the year 1970. This format is an international standard and may be found in website statistics and timestamps from various programs (e.g.: 1394790117.49).
- **Excel Timestamp (days)**: This format is used as an internal representation of dates in MS Excel and indicates the number of days which have passed since January 1, 1900. The time of the day is indicated by decimals (e.g.: 41712.40414).
- **German:** This format is the usual date format in middle Europe. It has the form: dd.mm.yyyy (e.g.: 14.03.2014).
- **Anglo-American:** This format is used in Anglo-American countries and has the form: mm/dd/yyyy (e.g.: 03/14/2014)

Format of output variable
Just like the input format you may also specify the output format. When both formats are selected, the text output at the right hand side of the interface shows an example transformation.

For the output format, one additional format is available:

- **Time of day**: The time of day has the form: hh:mm:ss (e.g.: 10:41:57)

Variable level
If the output format is numerical you may choose whether the timestamp is to be stored as integer or floating point number. If you choose integer values, the manner of transformation may be chosen. The integer may be created by cutting of the decimals (rounding down) or by algebraic rounding where 0.5 is rounded up to the next integer.

Cutting off decimals may be useful for the excel timestamp where the integer names the day and the decimals indicate the time of day. Here, algebraic rounding would transform 1pm on August 1 to August 2 which may not be intended.

Output File
The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

7.2. Transform Scale to Groups

Another transforming function may be used to divide continuous scale variables to group variables. This may prove useful to partition timestamps in longitudinal data to phases or to partition numeric variables to quantiles.

Applications:

- Grouping of high-frequency timestamps to intervals
- Splitting normally distributed variables to quantiles
- Detecting confidence intervals and tails of distributions
- Detecting peaks in time series data

Options:

Input Table
This file contains a table with at least one scalar variable.

Scale Variable
From a list of all variables in the dataset you may choose the variable you wish to divide to groups. The variable should be metric and contain more different values than the groups you wish to create. Missing values result in missing group identification.

Output Variable
The transformation results in a new variable. You may enter any name for this variable.

Group Mode
There are different methods you may use to divide the scale variable. The optimal method depends on the type of the variable and the purpose of grouping (see Figure 7).

- **Equally sized groups**: This method divides the continuous variable to equally sized groups (quantiles). You may select four different options for this method:
    - Median: Two groups with equal amounts of cases will be produced

- Tertiles: Three groups with equal amounts of cases will be produced
- Quartiles: Four groups with equal amounts of cases will be produced
- Quintiles: Five groups with equal amounts of cases will be produced

- **Equal steps between groups**: This method divides the continuous variable to groups with equal distance. You may set the steps manually by entering an integer or decimal value.
  - **Size of step:** You may enter the size of the steps manually. The steps may be entered as integer or floating point numbers.
- **Tails of the distribution**: This method divides the continuous variable to three steps, labeled -1, 0 and 1. The -1 group contains the lower tail of the distribution. The 1 group contains the upper tail of the distribution. The 0 group contains all values between the tails (the confidence interval). You may choose the size of the confidence interval.
  - **Size of the tails:** You may choose the size of the confidence interval and thus the size of the tails. The confidence interval and tails are always two-sided. This means that a confidence interval of 95% results in two tails each hilding 2.5% of the values.

Output file

The file in which the table with the new variable is written. The table corresponds ot the input table with one variable added at the end of the dataset.

Labels of the groups:

The labels of the groups depends on the method used for grouping.

- For quantiles the groups are numbered beginning with 1.
- For equal steps the groups are labeled with the lowest value within the group.
- For tails, the tails are labeled -1 and 1 while the confidence interval is labeled 0.

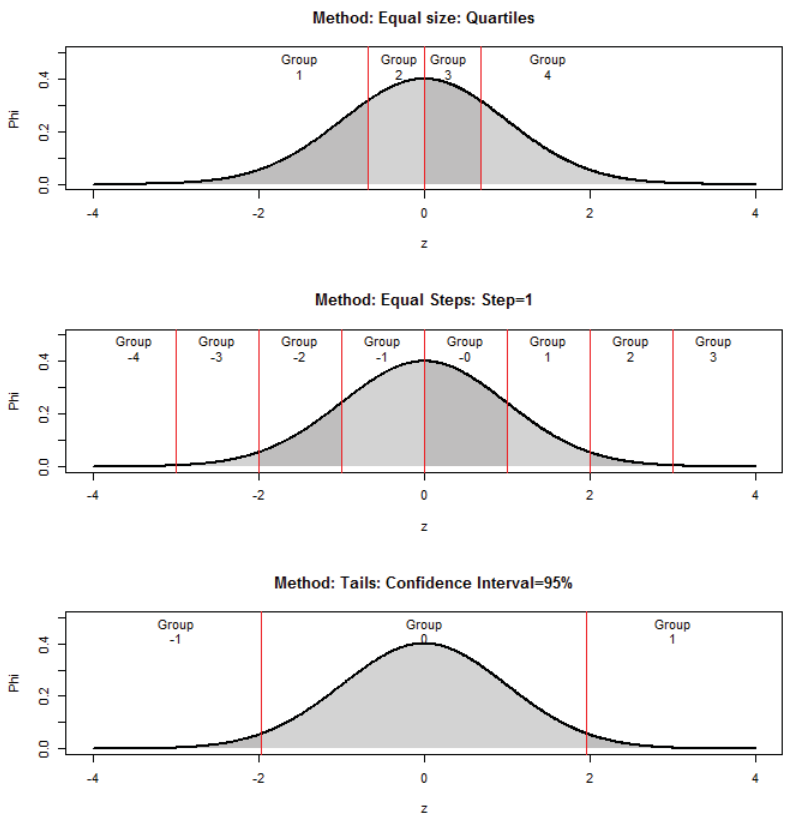


Figure 7: Visualization of the methods for grouping continuous variables. In this example, a normally distributed variable (standardized) was used. Note the labelling scheme for the different types of groups. For quantiles the groups are numbered, for equal steps the groups are labeled with the lowest value of the group. For tails, the tails are labeled -1 and 1 while the confidence interval is labeled 0.

Chapter 8. Co-Occurrence Analysis

Similar to dummy transformations, co-occurrence analyses are used to investigate the occurrence of values of a multinomial variable within groups defined by a second variable. Different from dummy transformations, however, the result is an NxN-Matrix providing information on how many times (or with which probability) two different values appear within the same group.

The starting point for co-occurrence analyses may be two multinomial variables as for dummy transformations or an already existing dummy table.

8.1. Co-Occurrence analysis from multinomial variables

Co-occurrence analyses of a multinomial variable require two different variables. First, you need to specify the group variable which contains the names of the groups. Then you have to indicate the multinomial variable whose values are tested for co-occurrence. You may then select different methods of how to express the co-occurrence of values within groups.

Applications:

Co-occurrrence analyses may be used to answer questions of the form: "(How frequently) do two values of this variable co-occur within groups defined by another variable?"

Using the example data, one might ask:

- How many times do the actors appear in the same issue of a TV show?
- How probable is the discussion of different issues on the same day?
- How likely is it that different arguments appear within the same statement of an actor?
- See Example: *How probable is the discussion of certain different issues on the same show?*

Options:

Input Table

The input table is a file which contains a table with at least two variables. As in dummy transformation, a grouping variable and a multinomial variable are required.

Group Variable

From a list containing all variables in the dataset you may choose the variable which defines the groups within which the co-occurrence of elements should be analyzed.

Multinomial Variable

From a list containing all variables in the dataset you may choose the variable which holds the elements that should be analyzed.

Method for calculating co-occurrence

For co-occurrence analyses a set of methods is available. These methods may be used to quantify the co-occurrence as measures of either closeness or distance:

- **Dichotomous**: The cells contain either 1 (these values co-occur in any group) and 0 (these values never co-occur)
- **Count**: The cells contain the exact number of groups the values co-occur in.

- **Probability of co-occurrence**: The cells contain the probability of co-occurrence measured as the product of the conditional probabilities of value A occurring in groups where B is presend and vice versa (P(A|B)*P(B|A)).
- **Percent Agreement**: The cells contain the number of groups where either both values are present or absent. Caution: For rare values this value is very close to 1 as they both don't appear in most groups.
- **Inverse Probability** of co-occurrence: The cells contain the negative logarithm of the probability of co-occurrence. This results in a measure of distance ranging from 0 (complete co-occurrence) to very large numbers. In order to prevent log(0)-errors, no co-occurrence is changed to 1 instance of co-occurrence in a dataset with twice the number of groups than present in the current data.
- **Sokal-Distance**: The cells contain the sokal-distance of two values which ranges from 0 to 1. 0 means perfect agreement, 1 means no co-occurrence.
- **Euclid Distance**: The cells contain the euclidian distance of two values.

Minimal number of elements per group

You may select the minimal number of elements each group has to contain. Groups containing less elements are removed from the dummy table. This means that the sum of values in each row is at least as high as the number you select here.

Note that the sum of a row containing dichotomous values is naturally lower than the sum of a row containing count data. Depending on the method you may have to adjust the minimal number of elements.

Minimal number of occurrences per value

You may select the minimal number of occurrences for each value of the multinomial variable to justify a dummy variable. Values occurring less than the specified number are removed from the table. This means that the sum of each column is at least as high as this number.

As the minimal number of elements per groups and the minimal number of value occurrences have to be fulfilled at the same time, sparse rows and columns are removed until both requirements are met.

Note that the sum of a column containing dichotomous values is naturally lower than the sum of a column containing count data. Depending on the method you may have to adjust the minimal number of elements.

Smallest Space Analysis

When choosing a method resulting in a distance measure (inverse probability, Sokal, Euclid), an additional output is possible. You may choose to export an R-script which calculates and displays the results of a Smallest Space Analysis (SSA) of the elements. In this analysis, the distances between each pair of elements is rescaled to two dimensions and visualized in a scatter plot.

The R-script may be executed in the statistical software project R using the package 'smacof'.

Output File

The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

8.2. Co-Occurrence analysis from Dummy-tables

As an input for co-occurrence analyses you may also use pre-existing dummy tables. The co-occurrence of true values (1) within the cases is calculated if you choose to use this procedure. All methods are equal to the co-occurrence analysis using a multinomial variable.

Applications:

- See Example: *Which arguments are often mentioned together?*

Options:

Input Table
The input table is a file which contains a table with dummy variables.

Dummy Variables
From a list containing all variables in the dataset you may choose the variables which contain the dummy values. Since the dummies are regarded as dichotomous variables (present or absent in this case) all types of variables are transformed to dichotomous variables. For numerical values, zeroes and missing values are considered 'absent' while numbers are regarded as 'present'. For string variables, any value other than empty strings and missing values are considered 'present' while all missing values are 'absent.

Invariate variables may not be chosen.

Method for calculating co-occurrence
For co-occurrence analyses a set of methods is available. These methods may be used to quantify the co-occurrence as measures of either closeness or distance:

- **Dichotomous**: The cells contain either 1 (these values co-occur in any group) and 0 (these values never co-occur)
- **Count**: The cells contain the exact number of groups the values co-occur in.
- **Probability of co-occurrence**: The cells contain the probability of co-occurrence measured as the product of the conditional probabilities of value A occurring in groups where B is presend and vice versa (P(A|B)*P(B|A)).
- **Percent Agreement**: The cells contain the number of groups where either both values are present or absent. Caution: For rare values this value is very close to 1 as they both don't appear in most groups.
- **Inverse Probability** of co-occurrence: The cells contain the negative logarithm of the probability of co-occurrence. This results in a measure of distance ranging from 0 (complete co-occurrence) to very large numbers. In order to prevent log(0)-errors, no co-occurrence is changed to 1 instance of co-occurrence in a dataset with twice the number of groups than present in the current data.
- **Sokal-Distance**: The cells contain the sokal-distance of two values which ranges from 0 to 1. 0 means perfect agreement, 1 means no co-occurrence.
- **Euclid Distance**: The cells contain the euclidian distance of two values.

Smallest Space Analysis
When choosing a method resulting in a distance measure (inverse probability, Sokal, Euclid), an additional output is possible. You may choose to export an R-script which calculates and displays the results of a Smallest Space Analysis (SSA) of the elements. In this analysis, the distances between each pair of elements is rescaled to two dimensions and visualized in a scatter plot.

The R-script may be executed in the statistical software project R using the package 'smacof'.

Output File
The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

Chapter 9. Social Network Visualization

Similar to tables created in dummy-transformations, Nogrod may create source files for social network visualizations in Visone (www.visone.info). These source files may then be opened and analyzed in Visone. The creation of these files is especially useful to illustrate the relation between variables.

For a social network structure, three sources of information are required. First, the subjects (nodes from which edges issue) and objects (nodes to which the edges point) have to be defined. Each of them may be stored in a multinomial variable. Second, there has to be some relation between the subjects and objects which may either be provided by a third variable indicating the relation or just be inferred from the number of times the subjects and objects co-occur within a case.

9.1. Social Network Visualization from two variables

In the standard case you have two variables (similar to dummy and co-occurrence analyses) which you would like to visualize as subjects and objects of a social network. A thir variable may be used to quantify or specify the relation between these variables.

Applications:

Social network visualizations answer questions of the type: "What kind of relation is there between values of different variables in this data?"

Based on the example dataset, one might ask:

- Who attacks whom?
- Who talks about which issues?
- Which shows cover which issues?

Options:

Input file
The input file contains a table with at least two multinomial variables defining the subjects and objects of the social network. A third variable may be used to quantify/identify the relation between subjects and objects.

Subject Variable
From a list containing all variables of the dataset you may choose the variable which holds the subject identifiers. In the social network the subjects are the nodes from which arrows issue.

Object Variable
From a list containing all variables of the dataset you may choose the variable which holds the object identifiers. In the social network the objects are the nodes toward which arrows point.

Identical values for subjects and objects
Choose whether the subjects and objects of the social network are coded by the same rules. If you choose 'yes' on this option, the subjects and objects are treated as one group of entities instead of two separate groups. This does not influence the way the network is visualized.

Relation variable

From a list containint all variables in the dataset you may choose the variable which defines the relation between subjects and objects. The relation may be numeric (e.g.: valence or count) or multinomial/string (e.g.: Type of relation).

Method for relation

The relation between subjects and objects may be calculated from the mere number of co-occurrence or from a third variable defining the relation.

- **Count**: The relation is the number of times a subject co-occurs with an object.
- **Inverse Count**: The number of times a subject and object co-occur (N) is inverted by the formula: 9/(1+N) which results in a distance of 9 for no co-occurrence and low distances for frequent co-occurrence.
- **Dichotomous**: The relation may only take the values 1 (present) and 0 (absent)
- **Type**: For multinominal relation variables, this method uses the last value of the relation variable for each pair as a value of their relation.
- **Mean Value**: Only if the relation variable is metric, this method will take the mean value of the relation variable for each pair as their relation.
- **Sum**: For metric relation variables only, this method calculates the sum of all relations.

Minimal occurrence

You may choose a minimal number of times a subject or object has to be present in the data to be counted in the analysis. Subjects and objects which occur less than the specified number of times are excluded from the visualization.

Output file

The file in which the adjacency matrix should be written. This file may be opened in Visone afterwards.

Output of node counts

The file in which a list of all nodes (subjects and objects) is written. For each node the number of occurrences is written in this output. You may merge this file with additional information on the nodes and import them in Visone.

9.2. Visone for Nino Abzianidze

As a special case, visualizations may also be generated from the data produced by Angrist in the content analysis of the project "Democratization and Nationalist Media: A Path to Civil Conflict" by Nino Abzianidze. In these data, the relation between senders and addressees of political appeals are stored on the level of actors in a special format.

This method is only applicable to the data in the table '_actors.txt' produced by Angrist in this project. Using this data, you may create source files for each kind of appeal.

9.3. Handling Visone

The software which may be used to visualize the networks is obtainable at: http://www.visone.info/ it is freely usable for academic purposes. For a quick introduction, the data from the example above is used to demonstrate the basic functions.

Open an adjacency matrix

To open a matrix, start visone and select 'open...' from menu 'file'. You may also click the blue folder icon to open a file. In order to read adjacency matrices, select 'adjacency matrix files (.txt, .csv)' as type.

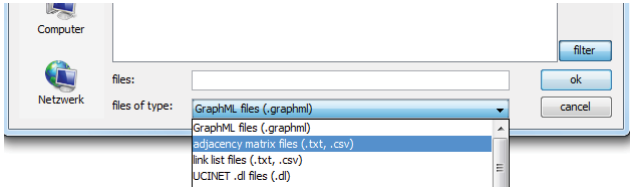


Figure 8: Setting the file type.

You may now open the file stored as adjacency matrix in the example above. In the dialog turning up, select the options as seen here:

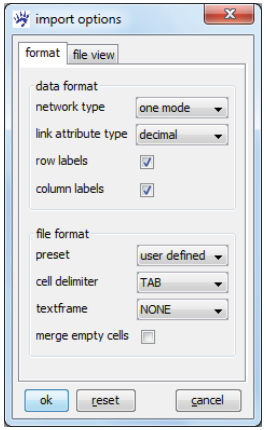


*Figure 9: Options for opening adjacency matrix files from Nogrod: The **network type** may be 'one mode' or 'two mode'. The difference lies in the role a node may take. In 'one mode', a subject may be the object of another subject. In 'two mode' a node may be either an object or a subject. For the data in the above example, one mode is the logical choice as Obama is both subject and object.*
*The **link attributes** are numbers (count of co-occurrence), so 'decimal' is a useful choice.*
*The table contains **labels for rows and columns** as seen in table 6.*
The file format has tabulators as separators and no quotation marks for text. Empty cells actually never occur in adjacency matrices from Nogrod but if they occur anyway they should not be merged. Merging would destroy the format of the table.

Opening the file with these settings results in a blind (i.e.: not labelled) social network.

Figure 10: Blind social network as created upon loading an adjacency matrix.

You may now add labels to the network by opening the attribute manager. This may be done only by clicking the item with yellow bars.

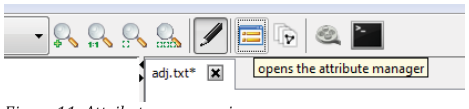


Figure 11: Attribute manager icon.

The attribute manager may be used to view, edit, and display values of all nodes and links. It has four options: 'configuration', 'values', 'operations' and 'import&export'.

Under 'values' you may see and eventually edit the values of each node and link. At first, only the row/column names are stored as values for nodes (id). The links have two values: An id, which is a running number, and a csv-value which is the value of the link in the adjacency matrix.

Under 'configuration' you may create new variables for links and nodes by entering a new name to the cell containing an asterisk (*). You may also select one of the variables to be used as label in the graphic display. Select 'id' as label for the nodes to display the names of the actors. Confirm by pressing 'apply' at the bottom of the attribute manager. You now see the names of all actors displayed as labels in the graphic display.

Under 'operations' you may transform values at need. In this analysis, no operation is required, however.

Under 'Import&export' you may import new information on nodes and links. Here you may import the number of times each node appeared (node counts) which are stored in the second file. Just select 'import' and browse for the file. Confirm by pressing 'apply'. When you return to 'values' you see a second column labelled 'Count'. Here, the number of times each actor appeared (either as subject or subject) is noted.

Optimize the layout of your network

Even with the labels the network still looks bare and far from appealing. The nodes and links are of a drakish grey shade and much too small to contain the whole labels. It is therefore reasonable to add some cosmetics.

To change the appearance of the nodes you first select 'select all' from the menu 'nodes'. Then you pick 'properties..' from the same menu. You now see a small dialog with three tabs (general, label, attributes) with which you may change the appearance of the nodes. In the tab general you may change the shape of the nodes to rounded rectangles with a larger width to engulf the whole labels and a lighter color. In the tab 'label' you may then change the font and size of the labels. The tab 'attributes' is useless when all nodes are selected. When only one node is selected you may see all attributes of this node in that tab.

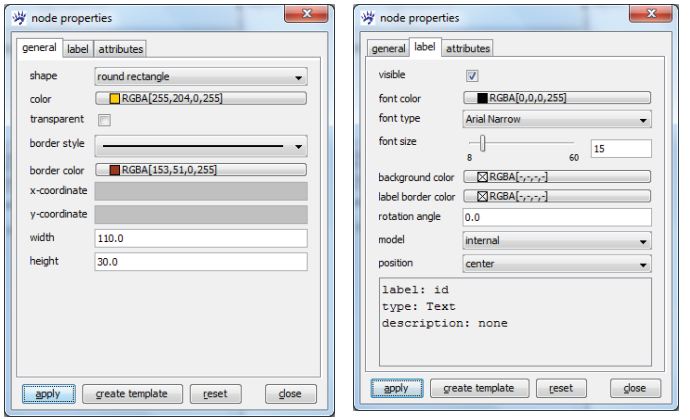


Figure 12: Node properties

Choosing the options as shown in figure 9, the network has a brighter and more friendly appearance (Figure 13). You may also change the shape and color of the links (arrows) with a similar properties dialog.

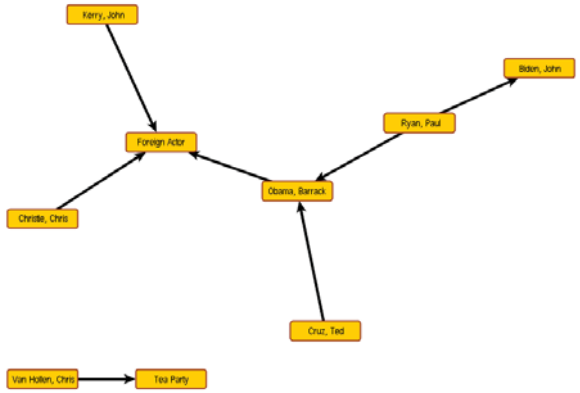


Figure 13: Appearance of the network after changing node and link properties.

If required you may also change the size or colour of nodes and the width or color of links according to their values. Remember that the number of times each node was present in the data is stored as an attribute 'Count' in this network.

To change the layout according to attributes, you may select the tab 'visualization' on the left-hand side of the visone window. In this tab you may change the layout and appearance automatically.

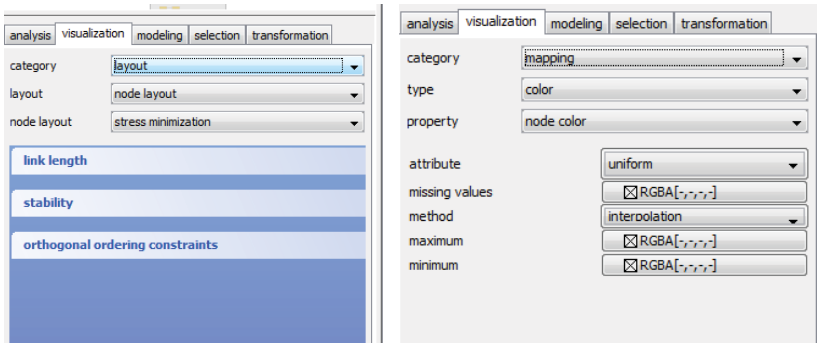
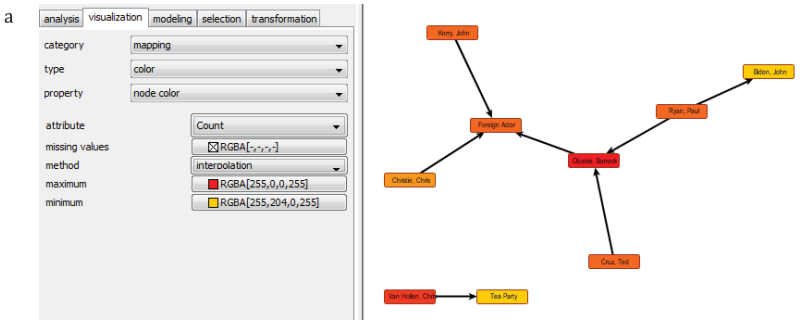


Figure 14: Visualization tab.

First, select 'mapping' from the dropdown menu 'category' the tab now changes its appearance and you may enter different parameters. Assume that wa want to change the colour of the nodes according to the number of times they appear in the data. Frequent nodes should be reddish, infrequent ones yellow (Figure 15a). Likewise, you may want to change the height of nodes according to their frequency (Figure 15b). You may also change the width of the links to the number of times an attack was present. This was stored in the adjacency matrix and has been imported to the network as 'csv value' (Figure 15c). In the current network, this has no effect, however, as all attacks are only present once.



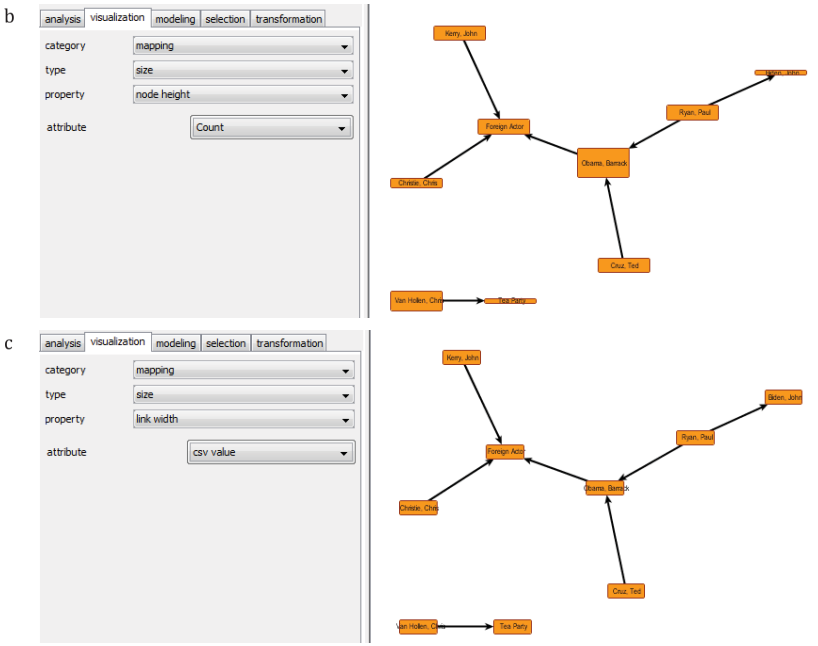
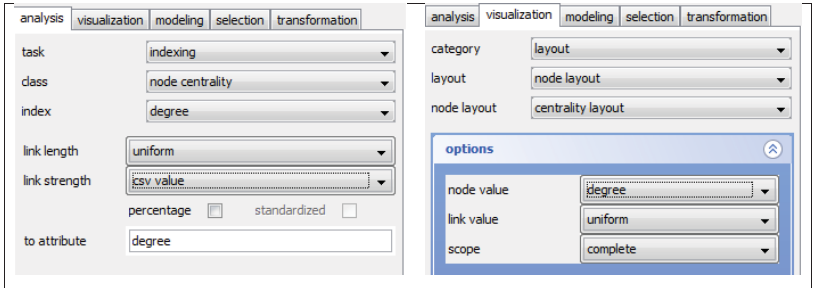


Figure 15: Data dependent properties of network appearance.

Network analysis

Using the tab 'analysis' on the left hand side of the window you may analyze the network. One useful procedure, which is present at the beginning, is the centrality calculation. Using this procedure, visone calculates a value for the centrality of nodes. The more links a node has to other nodes, the more central it is. When centrality is calculated, you may display the result using the 'layout' category of the 'visualization' tab (Figure 16).



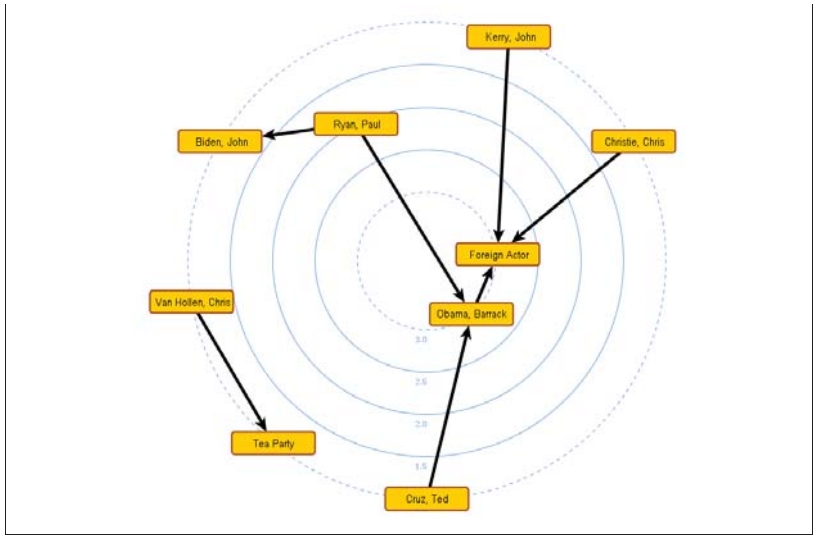


Figure 16: Calculate and display node centrality.

Chapter 10. Time Series Analysis

Time series analysis procedures allow the preparation and transformation of time series data. Time series transformations require a continuous numeric variable which is used as a timestamp.

10.1. Detect Peaks

Peak detection is a function which may be used to identify positive and negative peaks of time series data. Peaks are defined as the highest point of the time series which lies outside a confidence interval which may be chosen by the user. For all regions of the curve which lie outside the confidence interval, the highest (or lowest) point is detected.

A variable is added to the dataset which has the value 0 for most of the cases and assumes the value 1 for positive peaks and -1 for negative peaks. Values are considered to be a peak of the function if they lie outside a defined confidence interval and are higher or equal to both of their neighbors. This means that only the points of high peaks are given the values 1 or -1.

Applications:

- See Example: *Finding peak skin conductance during a movie*

Options:

Input Table

This file contains a table with time series data. It has to caintain at least one variable holding timestamps and another variable containing a time series of measurements for each timestamp.

Time Variable

From the complete list of variables in the dataset you may select the variable which holds the timestamps. Timestamps have to be numeric (seconds, hours, days...). If the timestamps have another format, use the function *Transform Timestamps* to transform timestamps to numeric format.

Each timestamp should be unique. If it is not unique, the last case bearing a certain timestamp will be used for calculation.

Measurement Variable

From a list of all variables you may select the variable holding the time series data. The variable has to be numeric and should be scaled and approximately normally distributed. Multinomial data may be read but result in nonsense results.

Output Variable

This function creates a new variable holding peak information. This variable has the value 0 for most values. For the highest point of positive peaks, the variable has the value 1, for the lowest point of negartive peaks it has the value -1.

Detect Peaks

You may choose which peaks you wish to detect. Positive peaks are values of the measurement variable lying above the confidence interval. Negative peaks are values which lie below the confidence interval.

Peak Threshold

As peaks are regions of the curve which lie outside a confidence interval. You may choose the width of the interval. You may also choose to find all peaks without any confidence interval. In this case all local maxima and minima of the curve are treated as peak. For curves containing noise this is not recommended.

Output File
The file in which the table with the new variable is written. The table corresponds ot the input table with one variable added at the end of the dataset.

10.2. Flatten moving Average

For time series data with moving averages, the detection of peaks or sudden changes in the average may be hard to detect. To amend for moving average, time series data may be decomposed to long-term moving averages and short-term peaks using this function.

The function takes time series data, a timestamp variable and an entered width for the gliding window to calculate the moving average. It then decomposes the time series data to a smooth curve of mean values and a flat curve of deviations from this mean curve. Depending on the size of the window, the variance of the time series data is distributed to the smooth curve or the deviations. For large windows, the smoothened curve exhibits less variation and the deviations exhibit more variation.

Applications:

- Smoothen a curve containing high-frequent noise
- Find peaks in a curve with moving average (in combination with procedure 'Find Peaks')
- Subtract the moving average from an unstable curve to obtain a curve with average 0
- See Example: *Find peak increases in skin conductance*

Options:

Input Table
This file contains a table with time series data. It has to caintain at least one variable holding timestamps and another variable containing a time series of measurements for each timestamp.

Time Variable
From the complete list of variables in the dataset you may select the variable which holds the timestamps. Timestamps have to be numeric (seconds, hours, days...). If the timestamps have another format, use the function *Transform Timestamps* to transform timestamps to numeric format.

Each timestamp should be unique. If it is not unique, the last case bearing a certain timestamp will be used for calculation.

Measurement Variable
From a list of all variables you may select the variable holding the time series data. The variable has to be numeric and should be scaled and approximately normally distributed. Multinomial data may be read but result in nonsense results.

Width of the gliding window
The moving average is computed using a gliding window with its center at the current position. You may specify the width of the movind window by entering a number. The units of the gliding window are the units of the time series variable.

Examle: If your time series data is available at 0.2 second steps and you enter a width of 3 seconds the average value at second 2.0 is calculated from all values between second 0.5 and 3.5.

Moving average Variable
This function creates a new variable holding the moving averages for each point in the time series. You may enter a name for this variable.

Peak Variable

This function creates a new variable holding the deviation from the moving average at each point of the time series. Adding this variable to the moving average variable, results in the exact value of the measurement variable at each point. The values in this variable indicate abrupt rise or decrease of the measurement variable.

Output File
The file in which the table with the new variable is written. The table corresponds ot the input table with one variable added at the end of the dataset.

10.3. Create Gliding Window

The gliding window procedure creates a gliding window on a time series dataset by copying cases to a number of groups. This procedure may be used to prepare a time series to aggregation.

The function creates a new dataset containing multiple copies of each case within the input table which may result in large files. Please be careful when handling large input files and consider using subsets for this kind of data transformation.

Options:

Input Table
This file contains a table with time series data. It has to caintain at least one variable holding timestamps and another variable containing a time series of measurements for each timestamp.

Time Variable
From the complete list of variables in the dataset you may select the variable which holds the timestamps. Timestamps have to be numeric (seconds, hours, days...). If the timestamps have another format, use the function *Transform Timestamps* to transform timestamps to numeric format.

Each timestamp should be unique. If it is not unique, the last case bearing a certain timestamp will be used for calculation.

Width of the gliding window
The moving average is computed using a gliding window with its center at the current position. You may specify the width of the movind window by seelcting a number. The units of the gliding window are the units of the time series variable.

Position of the identifier
A new variable will be created which contains the identifier for the gliding window each case belongs to. Since a case may belong to many gliding windows, each case is copied for each window it belongs to. The identifier of the gliding window is a value from within the window. You may choose which value should be used. You may choose the first element to name each window with the first time stamp which still belongs to the window. Respectively, you may choose the name to be the middle or last timestamp.

Position Variable
This function creates a new variable holding the identifier of the gliding window. You may enter any name for this variable.

Output File
This is the file in which the table including the new variable is written. The table corresponds to the input table with one variable added at the end of the dataset and a multidude of copies for each case.

Chapter 11. Sequence Analysis

As a special case of time series analysis, Nogrod also contains a function to analyze a seqence of data to find patterns within the sequence for further analysis.

11.1. Find common sequences

The function for finding common sequences allows for the identification of sequence patterns within long sequences of data. This may be useful for time series data of nominal variables which indicate the state or actions of a person at each given time. As an example, the series of fixations of areas of interest in eye-tracking studies, or the sequence of issues in a news show may be investigated using this function.

The function is currently limited to the identification of sequences of four values.

Sequences with omissions:

The function does not only find coherent sequences but may also identify sequences with one omission. This means that there are four different sequences wich may be found in a sequence of length 4. In the sequence A-B-C-D, the first sequence of length three is ABC, the second is BCD. The fourth allows for one omission in the second place and is ACD. The fourth allows for one omission in the third place and is therefore ABD.

Keep in mind that the allowance of omissions lead to much larger data output.

Nomenclature of output:

Output of this function is a table containing all sequences starting from each case in the data. The sequences are thereby named according to the following system for a sequence A-B-C-D-E:

- Seq_2: AB, BC, CD, DE
- Seq_2a: AC, BD, CE
- Seq_3: ABC, BCD, CDE
- Seq_3a: ACD, BDE
- Seq_3b: ABD, BCE
- Seq_4: ABCD, BCDE
- Seq_4a: ACDE
- Seq_4b: ABDE
- Seq_4c: ABCE

The sequences denominated with a suffix -a, -b, and -c are sequences with one omission.

Depending on the mode of output, the sequences are written in one variable each which is named after the sequence (broad table), or as one list of sequences in one common variable with a second variable indicating the type of the sequence (long table).

Options:

Input Table
This file contains a table with sequence data. It has to caintain at least one variable containing the sequence of interest. It may also contain a time variable and a variable differentiating between different cases for which this sequence data is available. If a time variable and a case variable are inside, it does not have to be sorted. If only the sequence data is available, it should naturally be in correct temporal order.

Sequence Variable

The one column in the data containing the sequence of interest. The sequence variable may be any list of nominal values.

Time Variable
If desired, the sequence may be sorted by a time variable in the data. The time variable should contain numerical values indicating the time for each value in the sequence.

Group Variable
When more than one sequence is present in the data, a group variable may be indicated which differentiates the sequences. This may be the person for which the sequence was recorded, for example.

Sequence Length
You may choose a length of 2, 3, or 4 values. The shorter sequences are automatically added, so that a length of 4 values also exports the sequences of langth 2 and 3.

Omission of one value
You may choose whether or not omission should be allowed. Be aware that omissions may double the output.

Output mode
You may choose whether the output table should be long (containing all sequences in one variable and multiplying the cases) or broad (equal amount of cases and one variable for each type of sequence).

Output File
This is the file in which the table including the new variable is written. The output table only contains the sequence variable, group variable, time variable and the sequences. All other information in the source table is not part of the output.

11.2. T-Pattern Analysis

The temporal pattern (T-Pattern) analysis was first introduced by Magnusson (2000) to identify patterns in the sequence of events in the presence of noise and slight variations. The basic idea behind the T-Pattern analysis is to find events which follow other events more often that would be expected in a random series of events. If any of these duplets are found in the data, they form events of themselves which may be part of a higher order pattern.

The result of the analysis is an account of found patterns, their number of occurrence and their location within the data.

In Nogrod, it is possible to perform a T-Pattern analysis for multiple parallel sequences and to take non-linear timestamps into account. This means that missing data or short bursts of high-frequency activity do not pose a problem to the analysis and that several sequences may be analyzed in one go to find common patterns in all of them.

Chapter 12. Cluster Analysis

As there is no statistical software yet which has an implementation of cluster analysis of count data with outlier omission, this function was added to Nogrod.

12.1. Cluster Analysis of Count data (HECATE)

This method for cluster analysis uses hierarchical bottom-up clustering to find homogeneous clusters of large quantities of items and rejects all items not belonging to any of the homogeneous clusters. The result thus contains only a part of the items entered to the analysis. All other items are omitted.

HECATE is a bottom-up hierarchical cluster analysis aggregating all elements and clusters to one sinlge hierarchical structure, using cluster centre distance to combine clusters. In the process of the routine, each step of aggregation is evaluated. If the step is harmful to the homogeneity of the solution, it is tagged for postprocessing. Upon completion ogf the tree, all harmful steps are undone to isolate homogeneous clusters from the tree.

The result of the analysis is a collection of homogeneous clusters which could not be linked to their nearest neighbor without scrificing their homogeneity. The condition for steps to be counted as hamful is thereby: If the combination of a cluster with center C and radius R to the nearest element with distance D results in a new cluster with radius R' exceeding the length of D, the step is considered harmful. This condition only applies if the next step would be the inclusion of an outlyer or another homogeneous cluster with a high distance to C.

The resulting set of clusters has been shown to exclude randomly distributed elements and other sources of noise in count data matrices (vgl. Wettstein, Submitted). It is therefore an ideal procedure for clustering natural language and content analysis data where some elements (stop words or frequent categories) have to be excluded from the solution.

Nogrod performs the cluster analysis and gives a series of outputs containing fine-grained documentation of the process, a dendrogram, and a dataset with cluster scores to be included in statistical software packages.

Options

Input Table

This file contains a table with multiple variables constituting a count data matrix. There may be other variables in the dataset as well, as the user is asked to manually select the variables constituting to the matrix.

Count Variables

From a list of all variables, you may select the ones you wish to include in the cluster analysis. In the text output, a summary of each variable is provided which includes the variable level and the number of missing values. For the cluster analysis, only variables with numerical data and without missing values should be included as cases with missing / string values are excluded.

Standardization

If desired, you may choose to standardize table prior to the analysis. The columns of the table will be normalized in the progress of the content analysis, in order to level out the influence of frequent elements, you may preprocess the table by standardizing the rows. It is recommended not to use standardization.

Output FIle

Specify the file in which the summary of results is written. This file will contain the contents and radius of each of the emerging clusters, as well as their mutual correlations.

Additional Outputs

There is a series of additional outputs which may be provided.

- Smallest Space Analysis: If you select this option, an additional output is created containing an R-Script to perform a SSA using the smacof package in R. The analysis provides a visualization of the elements and their distance in the final solution and may be used for exploatory qualitative analysis and interpretation of the output.
- Dendrogram: This output is again an R-Script to visualize the results. With this script, a dendrogram of the cluster analysis may be visualized to investigate the relative postion of emerging clusters.
- Detailed clustering history: In this output, all decisions and calculations of the clustering process are documented for debuggin purposes. If you don't trust the cluster solution, you may investigate this file.
- Cluster loadings for all items: In this output, the distance of all elements and all cases to the cluster centers is provided. You may use this table for additional analyses of second-order clusters in statistical software.
- Complete Table of cluster vectors: In this output, all vectors of the analysis are provided. You may use them for additional factor- and cluster analyses using different methods.

Reference:

Wettstein, M. (Submitted). Identifying Clusters amid Noise: Hierarchical Exploratory Cluster Analysis with Term Exclusion. *Communication Methods and Measures*

Chapter 13. Reliability Testing

As Nogrod was designed to handle large amounts of data, mainly from automated and manual content analysis, it also includes functions for testing interrater (or intercoder) reliability.

13.1. Test interrater reliability

The function for testing interrater reliability allows for extensive tests of interrater reliability across large amounts of units of analysis and large sets of coders. It includes a growing sample of coefficients and speical functions which enable quick and thorough analysis of coder reliability.

In order for this function to work properly, the input data should come in a standardized form where the coding of each coder for each unit of analysis occupies one row (See Table 4). The number of categories (columns) is not limited. Missing values within the coding of an unit of analysis are allowed, so are missing units of analysis for a coder or missing coders for a given unit of analysis. The function only compares the information which is present.

Table 4: Input format for reliability tests.

Unit of analysis	Coder	Genre	Relevance	Politics	Economy
text1	joe	2	1	1	0
text1	anne	2	1	1	0
text1	fred	2	1	1	1
text2	joe	3	1	0	1
text2	anne	3	2	0	1
text2	fred	1	1	0	1
text2	jim	3	1	0	1
text3	joe	2	3	1	0
text3	anne	2	3	1	0
text3	fred	2	3	1	
text3	jim	3	3	0	
text4	joe	1	2	1	1
text4	jim	1	2	1	1

The function is programmed to calculate the most widely used coefficients for interrater reliability. In this chapter, a short introduction to each of the coefficients is offered in order to understand their background and calculation. For further information on the coefficients, please refer to the references offered at the end of this chapter.

Pairwise comparisons:

The most basic coefficients are based on the pairwise comparison of all coders among each other or against a correct solution (gold standard) for the texts that were rated. In order to calculate the coefficients for pairwise comparisons, cross tables are created which hold the information on the agreement between each pair of coders. For the purpose of an example, the category 'Genre' from Table 4: Input format for reliability tests.Table 4Table 4: Input format for reliability tests. is used to demonstrate the calculation of pairwise comparisons.

The cross tables obtained from comparing all coders are depicted in Figure 17.

Anne × Joe

	1	2	3
1	0	0	0
2	0	2	0
3	0	0	1

Fred × Joe

	1	2	3
1	0	0	1
2	0	2	0
3	0	0	0

Jim × Joe

	1	2	3
1	1	0	0
2	0	0	0
3	0	1	1

Fred × Anne

	1	2	3
1	0	0	1
2	0	2	0
3	0	0	0

Jim × Anne

	1	2	3
1	0	0	0
2	0	0	0
3	0	1	1

Jim × Fred

	1	2	3
1	0	0	0
2	0	0	0
3	1	1	0

Figure 17: Cross tables for the variable 'Genre'.

From these crosstables, the pairwise agreement may be computed for each pair of coders (e.g.: Joe and Anne agree 100%, whereas Joe and Fred Agree by 66%). The most basic coefficient for pairwise comparison is calculated as the mean agreement across all coder pairs. This coefficient is called **Percent Agreement** or **Holsti Coefficient**. In this example, the coefficient would be 58.3%. For the Holsti coefficient, values above 80% are usually considered good agreement.

While the advantages of Holsti are its very easy calculation and interpretation, it has the disadvantage of being too liberal when it comes to unevenly distributed categories. In categories which have the same value in 90% of the cases, the chance of accidentally choosing the same value lies at 81%. Therefore, an agreement of 80% would not be considered very reliable in a real content analysis. In order to correct for this shortcoming, Kappa coefficients are often used in pairwise comparisons.

Kappa coefficients, such as **Cohen's Kappa** and **Soctt's Pi** are using the per chance agreement (Pc) of two coders, given the distribution of values, and indicate the distance of the true agreement (PA) versa the per chance agreement. In order to compute Kappa values, the following formula is used:

$$k = \frac{PA - Pc}{1 - Pc}$$

Kappa is exactly 0 when the actual agreement is equal as the per chance agreement. When the actual agreement is 100%, kappa equals 1. When the actual agreement is lower than the per chance agreement, kappa is negative. Values above 0.4 are generally considered acceptable, whereas values over 0.6 are considered good.

The only difference between Cohen's Kappa and Scott's Pi lies in the calculation of per chance agreement. For Cohen's Kappa, the per chance agreement is calculated individually for each pair of coders by summing up the suqared marginal chances for each category. In the above example, the per chance agreement for Joe and Fred would therefore be $(1/6)^2+(4/6)^2 + (1/6)^2 = 1/36 + 16/36 + 1/36 = 18/36 = 0.5$. With the actual agreement of Joe and Fred being 66%, Cohen's Kappa would equal: .166/.5 = 0.333.

For Scott's Pi, the per chance agreement is calculated by using the distribution of the values for the whole dataset. It is therefore constant for each pair of coders and allows for a direct comparison of pairwise coefficients. For Genre, the per chance agreement equals $(3/13)^2 + (6/13)^2 + (4/13)^2 = 0.361$. Therefore, the Scott's Pi for Joe and Fred would equal (0.666-.361)/(1-.361) = 0.478. In this case, Scott's Pi is slightly larger than Cohen's Kappa. In most cases, however, Cohen's Kappa is more liberal than Scott's Pi as the per chance agrement for a

badly agreeing pair of coders may be quite low. Scott's Pi is the most conservative and comprehensive coefficient for pairwise comparisons and should be preferred.

Note, that both Kappa coefficients are undefined for invariant variables because the per chance agreement of invariant variables equals 1.0. Therefore, kappa would result in a division by zero and may not be calculated. In Nogrod you may choose whether to set kappa for invariant variables to 1 (default) or a missing value.

Category comparisons

Slightly different from pairwise comparisons, variable coefficients such as **Krippendorff's Alpha** calculate the agreement on one category for any given team of raters. There are no pairwise comparisons in this calculation which means that there are no scores for pairs of coders or for single coders. The basic formula for behind Krippendorff's Alpha is the ratio of variance of a category within a unit of analysis and the overall variance of a category over all units of analysis.

To calculate Krippendorff's Alpha of M coders on N units of analysis, where the coding decision of Coder c on unit u is C(c,u) would therefore be:

$$\alpha = \frac{D_o}{D_e}$$

With D_o, being the mean distance within all units of analysis:

$$D_o = \frac{1}{N}\sum_{u}^{N}\frac{1}{M(M-1)}\sum_{c1}^{M}\sum_{c2}^{M} distance(C(c1,u), C(c2,u))$$

And D_e, being the mean distance between all coded values:

$$D_e = \frac{1}{N(N-1)+M(M-1)}\sum_{u1}^{N}\sum_{u2}^{N}\sum_{c1}^{M}\sum_{c2}^{M} distance(C(c1,u1), C(c2,u2))$$

The distance between two codes is thereby calculated differently for metric, ordinal and nominal variables.

- For nominal variables, the distance is 0 for equal values and 1 for different values.
- For ordinal variables, the distance is relative to the number of codings of the ranks between the ranks coded by the two coders. If they agree, the distance is 0.
- For metric variables, the distance equals the square distance of the values.

Coder comparisons

For large teams of coders, another coefficient has been introduced recently, which allows for the calculation of a score for each coder in the team. **Fretwurst Lotus** is computed as the mean agreement with the mode value of a category for a unit of analysis. This coefficient indicates in how many cases a coder is in agreeement with most of the other coders.

Since Fretwurst Lotus does not include the agreement among coders which do not agree with the mode value, it is very liberal and is higher than Holsti in most cases. It is almost impossible to reach a Lotus below 0.5 and in most cases it ranges from 0.8 to 1.0. Regression analyses have shown that Lotus is relative to Holsti with 94.6% of variance being explained by the formula Lotus = 0.363 + 0.643*Holsti.

Summary of coefficients:

Since all coefficients have their benefits and shortcomings, a short overview is offered here:

- Holsti: Easy to calculate but very liberal for unevenly distributed categories. Allows for comparison with one solution.
- Cohen's Kappa: Corrects for per chance agreement but hard to calculate and hard to interpret across large teams of coders. Allows for comparison with one solution.
- Scott's Pi: Easy to calculate and comprehensive for large teams of coders. Converges with Krippendorff's Alpha for large samples. Allows for comparison with one solution.
- Krippendorff's Alpha for nominal data: Slow computation but standard measure for content analyses in communication sciences.
- Krippendorff's Alpha for ordinal data: Very slow computation and hardly comprehensive.
- Krippendorff's Alpha for metric data: Slow computation but comprehensive and useful for rating data.
- Fretwurst Lotus: Very easy to calculate and gives information on the fit of single coders, but it is far too liberal to be taken serious.

It is therefore recommended to use Holsti, Scott's Pi and Krippendorff's Alpha together to get a comprehensive and detailed Analysis of interrater reliability.

References:

- Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, *20*(1), 37–46. doi:10.1177/001316446002000104
- Fretwurst, B. (In Press). Intercoderreliabilität und Expertenvalidität bei Inhaltsanalysen: Erläuterungen zur Berechnung des Koeffizienten Lotus mit SPSS. In. W. Wirth, K. Sommer, M.Wettstein, J. Matthes (Eds.). [Fortschritte in der Inhaltsanalyse].
- Krippendorff, K. (2004). *Content analysis: An introduction to its methodology*. Thousand Oaks, Calif: Sage Publ.
- Scott, W. A. (1955). Reliability of Content Analysis: The Case of Nominal Scale Coding. *Public Opinion Quarterly*, *19*(3), 323–325.

Applications:

This function has only one application: Testing the agreement of a team of coders on a series of units of analysis.

Options:

Input Table

This file contains a table with reliability test data. The data should be formated as seen in Table 4.

Unit of analysis

From all variables in the file you may select the variable which denotes the unit of analysis. In the example in Table 4 this would be 'Unit of analysis'.

Coder identifier

From all variables in the file you may select the variable which denotes the coder. In the example in Table 4 this would be 'Coder'.

Test Variables

From all variables (except for the identifiers for coder and unit of analysis) you may select the variables for which a reliability test should be performed.

Test Cases

From all units of analysis within your file you may select the units which are to be included in the test.

Compare Coders

From all coders within your file you may select the coders whose decisions you want to compare.

Core Coder

You may choose one coder to test all other coders against when computing percent agreement (Holsti), Scott's Pi or Cohen's Kappa. If you want to compare each coder to all other coders, select 'No core coder'.

Core coders may be used when one of the coders is the project leader to whom all other coders need to be compared to ensure the validitiy of the decisions of each coder.

Method

There are a number of coefficients which may be calculated in a reliability test. You may choose one or several of these coefficients to be calculated. See detailed descriptions for each coefficient above.

Special Options

You may choose some special options for the reliability test. These options are:

- No Kappa value for invariant variables: In the case of invariant variables which are assigned the same value for all units of analysis by all coders, the chance agreement is 100% and may not be beaten by human coders. You may choose to omit these variables from the analysis.
- Count missing values as coding: In some cases it may be required to count a missing value as a valid code. By default, missing values are considered missing decisions and are omitted from the analysis. If this option is checked, all missing values are included in the analysis.
- Demand a minimum of 2 / 5 comparisons: Especially for datasets with few units of analysis it may be useful to omit comparisons where only the decision on one unit of analysis is compared.
- RICO: When choosing a 'Reliability if coder Omitted'-Test (cf. Scharkow 2012: 129), all coefficients will be computed for the complete team of coders. Subsequently, for each coder in the team, the coefficients are calculated for the team without this coder. If the reliability increases by omitting a single coder, this coder is harmful to the overall relaibility and should be trained again.

Output File

The Output file contains a summary on the test options you entered and a table containing the mean values of all coefficients across all variables and coders.

Report File

The Report contains more information than the output file since it also contains individual tables with summaries on all coefficients which are calculated. All tables are stored with tabulator spacers. The file is best viewed by copy&pasting its contents to an empty Excel sheet.

Table 5: Result of the reliability test from the example data shown above. While the reliability is satisfactory for most variables and good for 'Relevance' and 'Economy', the reliability of 'Genre' is quite badly coded with an agreement <60% and Scott's Pi = 0.348.

	Alpha Metric	Alpha Nominal	Alpha Ordinal	Kappa	Lotus	PA	Pi
Economy	0.738	0.738	0.808	0.556	0.875	83.30%	0.64
Genre	0.374	0.583	0.374	0.198	0.834	58.30%	0.348
Politics	0.718	0.718	0.718	0.444	0.917	77.80%	0.531
Relevance	0.911	0.791	0.871	0.685	0.917	80.60%	0.696
Total	0.685	0.708	0.693	0.471	0.885	75.0%	0.554

Chapter 14. Examples

14.1. Add cases from another dataset

For this procedure to work, two datasets should be present. As we only have one example dataset, we do something stupid here as we are just appending the table to itself. To do this, perform the following steps:

1. Select 'Merge Datasets' -> 'Add Cases from another table' from the menu
2. Select your first table ('Example_data.txt'), it contains a header and is separated by tabstopps
3. Select your second table ('Example_data.txt'), it contains a header and is separated by tabstopps

The text output now informs you on the size of the two datasets. As they are both the same, both of them contain 14 variables and 19 cases.

The output also informs you on the congruence of variables. It found 14 variables which are present in both files and none which are present only in one file.

4. In the list you see all variables and the information whether they are present in both tables or only in one. You may select the variables you wish to have in the output. For this example, just select Actor, Date, Issue and Quotation. Select them on the list at the top and press 'Add Item' to choose them. Then confirm your selection.
5. You may now choose a name for the output file and its format. (e.g.: 'Double-Table.txt'). Upon confirmation, the tool merges the datasets.

The text output now informs you on the size of the output which has 38 cases and only four variables.

The variables in the output table have the same order as in the input tables. If the names of the variables differ in both tables (i.e. if some variables only appear in one table) the table first contains the variables from the first table and then the variables which were only in the second table.

Table 6: Merged dataset when adding the example table to itslef.

Date	Actor	Quotation	Issue	
Mar 13	Obama, Barrack	Direct	Budget	
Mar 13	Van Hollen, Chris	Indirect	Budget	
Mar 13	Ryan, Paul	Direct	Budget	
Mar 13	Cruz, Ted	Direct	Obamacare	
Mar 14	Christie, Chris	Indirect	Obamacare	
Mar 14	Ryan, Paul	Direct	Budget	
Mar 14	Kerry, John	Indirect	Budget	
Mar 20	Christie, Chris	Direct	Iran	
Mar 13	Van Hollen, Chris	Direct	Budget	
Mar 15	Obama, Barrack	Indirect	Iran	
Mar 15	Cruz, Ted	Direct	Iran	
Mar 20	Van Hollen, Chris	Direct	Budget	
Mar 20	Obama, Barrack	Indirect	Obamacare	
Mar 14	Ryan, Paul	Direct	Obamacare	
Mar 14	Kerry, John	Direct	Obamacare	
Mar 14	Cruz, Ted	Direct	Obamacare	
Mar 14	Van Hollen, Chris	Direct	Budget	
Mar 15	Obama, Barrack	Direct	Iran	
Mar 15	Kerry, John	Indirect	Iran	
Mar 13	Obama, Barrack	Direct	Budget	from here, the list repeats
Mar 13	Van Hollen, Chris	Indirect	Budget	
Mar 13	Ryan, Paul	Direct	Budget	

Mar 13	Cruz, Ted	Direct	Obamacare
Mar 14	Christie, Chris	Indirect	Obamacare
Mar 14	Ryan, Paul	Direct	Budget
Mar 14	Kerry, John	Indirect	Budget
Mar 20	Christie, Chris	Direct	Iran
Mar 13	Van Hollen, Chris	Direct	Budget
Mar 15	Obama, Barrack	Indirect	Iran
Mar 15	Cruz, Ted	Direct	Iran
Mar 20	Van Hollen, Chris	Direct	Budget
Mar 20	Obama, Barrack	Indirect	Obamacare
Mar 14	Ryan, Paul	Direct	Obamacare
Mar 14	Kerry, John	Direct	Obamacare
Mar 14	Cruz, Ted	Direct	Obamacare
Mar 14	Van Hollen, Chris	Direct	Budget
Mar 15	Obama, Barrack	Direct	Iran
Mar 15	Kerry, John	Indirect	Iran

14.2. Add information on the actors to the table

For this example, a second table is needed. We therefore use the data in the example dataset Example_Functions.txt (see Table 2).

This information may then be added to the example dataset.

1. Select 'Merge two datasets'.
2. Choose the example data 'Example_Data.txt' as main table. Confirm and select 'Example_Functions' as key table.
3. Now the tables have to be linked by key variables. Choose 'Actor' as the key variable in the main table by adding them to the selection list.
4. Choose 'Actor' as the key variable in the key table.
5. From the variables remaining in the key table (which is only one) you may select the ones to be added to the main table. Select 'Function'
6. Choose an output file and confirm.

The result of this transformation is a new table which contains an additional column (Table 7)

Table 7: Merged dataset (omitting some variables to save space) with the additional variable. The value has been added to all rows of the dataset, according to the actor on this row

Show_ID	Station	Date	Actor	Actor_ Affiliation	Quotation	Issue	Function
FN001	Fox	Mar 13	Obama, Barrack	Dem	Direct	Budget	President
FN001	Fox	Mar 13	Van Hollen, Chris	Dem	Indirect	Budget	Representative
FN001	Fox	Mar 13	Ryan, Paul	Rep	Direct	Budget	Representative
FN001	Fox	Mar 13	Cruz, Ted	Rep	Direct	Obamacare	Senator
FN002	Fox	Mar 14	Christie, Chris	Rep	Indirect	Obamacare	Governor
FN002	Fox	Mar 14	Ryan, Paul	Rep	Direct	Budget	Representative
FN002	Fox	Mar 14	Kerry, John	Dem	Indirect	Budget	Secretary of State
FN003	Fox	Mar 20	Christie, Chris	Rep	Direct	Iran	Governor
PB001	PBS	Mar 13	Van Hollen, Chris	Dem	Direct	Budget	Representative
PB002	PBS	Mar 15	Obama, Barrack	Dem	Indirect	Iran	President
PB002	PBS	Mar 15	Cruz, Ted	Rep	Direct	Iran	Senator
PB003	PBS	Mar 20	Van Hollen, Chris	Dem	Direct	Budget	Representative
PB003	PBS	Mar 20	Obama, Barrack	Dem	Indirect	Obamacare	President
AB001	ABC	Mar 14	Ryan, Paul	Rep	Direct	Obamacare	Representative
AB001	ABC	Mar 14	Kerry, John	Dem	Direct	Obamacare	Secretary of State
AB001	ABC	Mar 14	Cruz, Ted	Rep	Direct	Obamacare	Senator
AB001	ABC	Mar 14	Van Hollen, Chris	Dem	Direct	Budget	Representative
AB002	ABC	Mar 15	Obama, Barrack	Dem	Direct	Iran	President
AB002	ABC	Mar 15	Kerry, John	Dem	Indirect	Iran	Secretary of State

14.3. Select statements of democrats containing an attack

For this example, two conditions have to meet. The values of two different variables have to take a certain value for a case to be selected.

1. Select the procedure 'Extract Subset from Data' -> 'Select Cases' and open the example dataset.
2. Select both variables which contain information required to extract this subset ('Actor_Affiliation' and 'Attack').

Upon confirming, the tool offers additional information on the values of these variables. The text output informs that there are four different combinations of values of these variables within the dataset:

```
File contains 14 variables
File contains 19 cases
Found 4 values for the variables ['Actor_Affiliation', 'Attack']. First 20:
      ['Actor_Affiliation=Dem; Attack=0', 'Actor_Affiliation=Dem;
      Attack=1', 'Actor_Affiliation=Rep; Attack=0', 'Actor_Affiliation=Rep;
      Attack=1']
```

From these values you may now select the value combinations which should be present in the subset.

3. Select the value 'Actor_Affiliation=Dem; Attack=1' and confirm.
4. Choose an output file and confirm.

Table 8: Subset of the data containing only cases where a democrat attacked an other actor. The argument variables were left out to save space.

Show_ID	Station	Date	Actor	Actor_ Affiliation	Quotation	Issue	Attack_Target
PB002	PBS	Mar 15	Obama, Barrack	Dem	Indirect	Iran	Foreign Actor
AB001	ABC	Mar 14	Van Hollen, Chris	Dem	Direct	Budget	Tea Party
AB002	ABC	Mar 15	Kerry, John	Dem	Indirect	Iran	Foreign Actor

14.4. How many times was each actor cited on each station?

In this short example, we try to find out how many times each of the actors was cited in each of the three different stations in the analysis. In order to answer this question, the following steps should be taken:

1. Open Nogrod and on the first page select 'Aggregate data' -> 'Create Dummy Table'. Confirm with 'Check'.
2. You are now asked for the input file. Browse the folder to find the file 'Example_Data.txt'. The dataset has a header and is separated by tabstopps. Select these options and confirm.

The console offers you an overview on the data just loaded. It should contain 14 variables and 19 cases. If you happen to have selected a wrong separator, the program would warn you.

3. Select the group variable 'Station' and the multinomial variable 'Actor' for this analysis. Set the method to be used to 'Count' as we are interested in how many times each actor appears on each station.

The console now offers some information on the variables selected. It informs you on the number of groups and values which will be used in the analysis. In addition, it provides the names of the groups and values. If there are more than 20 groups or values, only the first 20 are displayed.

```
File contains 14 variables
File contains 19 cases

Case-Variable: Station
3 Cases in this File. First 20: ['ABC News', 'Fox News', 'PBS']

Value-Variable: Actor
6 Values for multinomial variable in this File. First 20: ['Christie,
      Chris', 'Cruz, Ted', 'Kerry, John', 'Obama, Barrack', 'Ryan, Paul',
      'Van Hollen, Chris']
```

4. Optionally, you may now restrict the number of groups and values by adding a lower limit of appearances for both. If you only want to include actors which appear at least three times to exclude all unimportant actors, you may set the limit for values to '3'. In this small dataset, however, this would not make much sense. So just leave it and confirm with 'Check'.
5. You may now select an output file, either by writing a filename to the textbox or by browsing and setting a filename. It is highly recommended to leave the ouptut format at a textfile with header (with variable names) and tabstopps. You may, however, change it at will. Confirm with 'Check'.

The program now calculates the values for the dummy table and writes them in the output file. This happens instantaneously in this example as there are only 19 cases to evaluate. For datasets with several hundred or thousand entries, it may take some minutes.

The result of this transformation is a textfile containing a crosstable of sources and actors (Table 2). The first column is always labelled as '#Group' and contains values identical to the values which were present in the group variable.

Note that the names of the variables are not necessarily well formed. They may contain blanks, commata and any other special characters which were present in the values of the multinomial variable used for dummy aggregation. This is no problem for further data processing in Nogrod but it may pose a problem when importing the data to statistical software packages which do not allow special characters in variable names.

Table 9: Output of Dummy Transformation

#Group	Actor_Christie, Chris	Actor_Cruz, Ted	Actor_Kerry, John	Actor_Obama, Barrack	Actor_Ryan, Paul	Actor_Van Hollen, Chris
ABC News	0	1	2	1	1	1
Fox News	2	1	1	1	2	1
PBS	0	1	0	2	0	2

14.5. Which issues were covered in each of the TV shows?

For this question, you do not want the number of cases each issue appeared but just the shows that presented them. For the most part, the routine is handled as in the last example:

1. Select the file 'Example_Data.txt' as source file with header and tabstopp.
2. Select the 'Show_ID' as group variable and 'Issue' as multinomial variable, select 'Dichotomous' as method.
3. Type the name of a file for output and hit 'Check'.

The result may be seen in Table 10. You get a Dummy-Table which contains the value 1 for each show an issue was covered in.

Table 10: Dummy Table indicating the appearance of issues in TV shows

#Group	Issue_Budget	Issue_Iran	Issue_Obamacare
AB001	1	0	1
AB002	0	1	0
FN001	1	0	1
FN002	1	0	1
FN003	0	1	0
PB001	1	0	0
PB002	0	1	0
PB003	1	0	1

14.6. Which were the most frequent actors and issues on each station?

Here, the data is to be summarized on station level (three groups) and the actors and issues are to be aggregated to the most frequent value.

1. Select 'Aggregate Data' -> 'Aggregate Cases' as a procedure and load the example data.
2. Select 'Station' as the group variable.
3. Add 'Actor' and 'Issue' as variables to be aggregated. Select 'Most Frequent' as method.
4. Choose an output file and confirm.

The result of this transformation is a new table with three cases (for the networks) and four variables (table 10). The first variable is the group variable, which is the TV station in this example. The second variable provides the number of cases which belong to this group and were aggregated. Then, the aggregated variables themselves follow.

Table 11: Result of aggregation of actor and issue to station level.

Station	Number_of_Cases	Actor	Issue
ABC	6	Kerry, John	Obamacare
Fox	8	Christie, Chris	Budget
PBS	5	Obama, Barrack	Budget

14.7. Reshape the dataset to one show per line

Here, the data is summarized to the level of shows and all variables are aggregated by reshaping them to a broad view. For demonstration purposes and graphic limits, only three of the variables (Actor, Actor Affiliation, and Issue) are used.

8. Select 'Aggregate Data' -> 'Aggregate Cases' as a procedure and load the example data.
9. Select 'Show_ID' as the group variable.
10. Add Actor, Actor_Affiliation, and Issue as variables to be aggregated. Select 'Transform Dataset to broad format' as method.
11. Choose an output file and confirm.

The result is quite a broad table as all variables have been multiplied four times to make space for all data which was aggregated. The resulting table contains four copies of each variable as there is a maximum of four speakers in the shows of the example data. The aggregated lines are added in a chronological order, so Actor01 of AB001 is Ryan, followed by Kerry, Cruz, and Van Hollen. For Groups with less than four cases, missing values are entered to unused variables. As FN003 only has one actor, this case has nine missing values which might be used to specify actor 02, 03, and 04.

Table 12: Result of the aggregation to broad format. The table is cut in two parts as it is too broad to be displayed on one line.

Show_ID	Actor01	Actor_ Affiliation01	Issue01	Actor02	Actor_ Affiliation02	Issue02
AB001	Ryan, Paul	Rep	Obamacare	Kerry, John	Dem	Obamacare
AB002	Obama, Barrack	Dem	Iran	Kerry, John	Dem	Iran
FN001	Obama, Barrack	Dem	Budget	Van Hollen, Chris	Dem	Budget
FN002	Christie, Chris	Rep	Obamacare	Ryan, Paul	Rep	Budget
FN003	Christie, Chris	Rep	Iran			
PB001	Van Hollen, Chris	Dem	Budget			
PB002	Obama, Barrack	Dem	Iran	Cruz, Ted	Rep	Iran
PB003	Van Hollen, Chris	Dem	Budget	Obama, Barrack	Dem	Obamacare

Actor03	Actor_ Affiliation03	Issue03	Actor04	Actor_ Affiliation04	Issue04
Cruz, Ted	Rep	Obamacare	Van Hollen, Chris	Dem	Budget
Ryan, Paul	Rep	Budget	Cruz, Ted	Rep	Obamacare
Kerry, John	Dem	Budget			

14.8. Which arguments were seen on which show?

For this example, the argument variables have to be aggregated on show level. Since the presence of an argument in one statement means that it was on the show, you should pick 'Maximum' as a method.

1. Select 'Aggregate Dataset' as a procedure and load the example data.
2. Select 'Show_ID' as the group variable.
3. Add all 'Argument_'-variables as variables to be aggregated. Select 'Maximum' as method.
4. Choose an output file and confirm.

The result is a table containing 7 variables. First, again, the group variable and the number of cases per group are provided. Then, the maximum values for all selected variables are shown. Please note, that the maximum values are returned as decimal numbers.

Table 13: Arguments appearing on each show

Show_ID	Number_of_ Cases	Arugment_ Moral	Argument_ Economy	Argument_ Voter	Argument_ Justice	Argument_ Faith
AB001	4	0.0	1.0	0.0	1.0	1.0
AB002	2	1.0	0.0	1.0	0.0	0.0
FN001	4	1.0	1.0	0.0	1.0	0.0
FN002	3	1.0	0.0	1.0	0.0	1.0
FN003	1	0.0	0.0	0.0	0.0	1.0
PB001	1	0.0	1.0	0.0	1.0	0.0
PB002	2	0.0	0.0	0.0	0.0	0.0
PB003	2	00.	1.0	0.0	1.0	0.0

14.9. How many arguments are there in the statement?

For this example, the sum of all argument values for each case have to be calculated.

1. Select 'Aggregate Data' -> 'Aggregate Variables' as procedure and load the example dataset
2. Select all argument variables (Hint: You may type 'argu' to the textbox above the list to narrow the list to the argument variables. Then hit 'Add all' to add all these variables). Use the method 'Sum'
3. Select a name for the output variable (e.g. 'Sum_Argument')
4. Choose an output file and confirm.

The result of this calculation is a new variable (e.g. 'Sum_Argument') which contains the sum of all argument variables.

Table 14: Result of the calculation of the number of arguments in each statement. The new variable contains decimal numbers and is attached to the end of the table (some variables were removed to save space).

Show_ID	Station	Actor	Argument_ Moral	Argument_ Economy	Argument_ Voter	Argument_ Justice	Argument_ Faith	Sum_ Argument
FN001	Fox	Obama, Barrack	1	1	0	0	0	2.0
FN001	Fox	Van Hollen, Chris	0	0	0	0	0	0.0
FN001	Fox	Ryan, Paul	0	0	0	1	0	1.0
FN001	Fox	Cruz, Ted	0	1	0	0	0	1.0
FN002	Fox	Christie, Chris	1	0	1	0	0	2.0
FN002	Fox	Ryan, Paul	0	0	0	0	1	1.0
FN002	Fox	Kerry, John	1	0	1	0	0	2.0
FN003	Fox	Christie, Chris	0	0	0	0	1	1.0
PB001	PBS	Van Hollen, Chris	0	1	0	1	0	2.0
PB002	PBS	Obama, Barrack	0	0	0	0	0	0.0
PB002	PBS	Cruz, Ted	0	0	0	0	0	0.0
PB003	PBS	Van Hollen, Chris	0	1	0	1	0	2.0
PB003	PBS	Obama, Barrack	0	0	0	0	0	0.0
AB001	ABC	Ryan, Paul	0	0	0	0	1	1.0
AB001	ABC	Kerry, John	0	0	0	1	0	1.0
AB001	ABC	Cruz, Ted	0	0	0	0	1	1.0
AB001	ABC	Van Hollen, Chris	0	1	0	0	0	1.0
AB002	ABC	Obama, Barrack	1	0	0	0	0	1.0
AB002	ABC	Kerry, John	0	0	1	0	0	1.0

14.10. Diversity of the networks with respect to actor affiliation

For this example, the group variable is the network and the multinomial variable in question is the actor affiliation.

1. Select 'Aggregate Data' -> 'Calculate Entropy' as a procedure.
2. Choose the example data as input table (tabstopps and header) and confirm.
3. Select 'Station' as group variable and 'Actor_Affiliation' as multinomial variable.
4. For the method you may select either option. Since both affiliations appear in all networks, the result will be the same. It would only make a difference for values which are unevenly distributed across groups.
5. Choose an output file and confirm.

The resulting table always contains three Variables. The first variable denominates the group (value of the group variable). The second variable indicates the number of cases within each group. The third variable gives the entropy as floating number.

For this example you may note that the entropy is highest in Fox as republicans and democrats are distributed 5:3. This is almost an equal distribution. It is, however, low for PBS where the ratio is 1:4.

Table 15: Output for the calculation of actor affiliation entropy across groups

Station	N_Elements	Entropy_Actor_Affiliation
ABC	6	0.9183
Fox	8	0.9544
PBS	5	0.7219

14.11. Actor Diversity across networks

As in the example above, the diversity within networks has to be calculated here. In this example, however, the values in each group differ slightly as not all networks have the same population of actors. This means that it will make a difference which method you use here.

1. Select 'Aggregate Data' -> 'Calculate Entropy' as a procedure.
2. Choose the example data as input table (tabstopps and header) and confirm.
3. Select 'Station' as group variable and 'Actor' as multinomial variable.
4. Choose one of the options.
5. Choose an output file and confirm.
6.

For this example both methods were used to compare the results. As you may see below, the entropy scores differ strongly depending on the method used. When using all actors as a population, FOX has the highest entropy as it shows all actors and prefers none. ABC has a slightly lower entropy as one actor (Chris Christie) is missing. The lowest entropy may be observed in PBS which has several missing actors.

When using only the actors within each group as population, the entropy scores are much closer together. Other than with the full population, missing actors do not count when using this method. This results in higher scores for all groups (except for FOX where all actors are present).

Table 16: Result for entropy of actors using all actors as population.

Station	N_Elements	Entropy_Actor
ABC	6	0.8710490
Fox	8	0.9671320
PBS	5	0.5887621

Table 17: Results for entropy of actors using only the actors within each group as population.

Station	N_Elements	Entropy_Actor
ABC	6	0.9697239
Fox	8	0.9671320
PBS	5	0.9602297

14.12. How probable is the discussion of certain different issues on the same show?

Here, the probability of co-occurrence of issues within shows is to be calculated.

1. Select 'Co-Occurrence for a Multinominal Variable' and open 'Example_Data.txt' (with header and tabstopps)
2. Select 'Show_ID' as group variable and 'Issue' as multinomial variable. Select Probability of Co-occurrence as method.
3. Set no required minimum for groups and values.
4. Select a file for output and confirm.

Table 18: Result of the co-occurrence analysis of issues within shows. While Iran is never mentioned in the same show with any of the other issues, Budget and Obamacare have a large overlap. They co-occur in 80% of cases. The values in the diagonal are always 10

	Issue_Budget	Issue_Iran	Issue_Obamacare
Issue_Budget	1	0	0.8
Issue_Iran	0	1	0
Issue_Obamacare	0.8	0	1

14.13. Which arguments are often mentioned together?

In this example, you do not need a group variable and a multinomial variable but just the dummies provided in the variables 'Argument_xx'. These five variables are already dummy variables which may be used for an analysis of co-occurrence.

1. Select 'Co-Occurrence Matrix from Dummy Table' as procedure and load the example dataset.

The tool now gives a text output informing you on the suitability of each variable as a dummy variable:

```
File contains 14 variables
File contains 19 cases
Show_ID: Not suitable as Dummy Variable. Invariant.
Station: Not suitable as Dummy Variable. Invariant.
Date: Not suitable as Dummy Variable. Invariant.
Actor: Not suitable as Dummy Variable. Invariant.
Actor_Affiliation: Not suitable as Dummy Variable. Invariant.
Quotation: Not suitable as Dummy Variable. Invariant.
Issue: Not suitable as Dummy Variable. Invariant.
Arugment_Moral: Numeric Variable: 15 / 4
Argument_Economy: Numeric Variable: 14 / 5
Argument_Voter: Numeric Variable: 16 / 3
Argument_Justice: Numeric Variable: 15 / 4
Argument_Faith: Numeric Variable: 15 / 4
Attack: Numeric Variable: 12 / 7
Attack_Target: Non-Numeric Variable: 12 / 7
```

From this output you can judge that most variables are not suitable as dummies as they are invariant. The Argument variables, as well as 'Attack' and 'Attack_Target' are suitable as they contain different values. The numbers (e.g. 15/4) indicate the number of 0 and 1 in the variables

(e.g.: Attack has 12 zeroes and 7 ones). Note, that for string variables all strings are counted as '1', whereas all missings are counted as '0'.

2. Select all Argument variables from the list and add them to the list of variables to be used as dummies. Choose 'Count' as method.
3. Select a file for data output and confirm.

Table 19: Co-occurrence of arguments in the same statement. The diagonal is always the sum of occurrences of each variable

	Argument_ Economy	Argument_ Faith	Argument_ Justice	Argument_ Voter	Arugment_ Moral
Argument_Economy	5	0	2	0	1
Argument_Faith	0	4	0	0	0
Argument_Justice	2	0	4	0	0
Argument_Voter	0	0	0	3	2
Arugment_Moral	1	0	0	2	4

Example: Who attacks whom?

In the dataset there is a variable naming the target of attacks for each statement. If there is no attack, the value of this variable is missing. This data may be used to create a social network.

1. Select the procedure 'Create Social Network Visualization' and open the example data (with header and tabstopps)
2. The subject of this network will be the actors from variable 'Actor'. The objects are the actors from 'Attack_Target'. The subjects and objects are coded by the same rule. This means that Obama, Barrack as a target of an attack is the same node as Obama, Barrack attacking a foreign actor.
3. You might set a lower limit of appearances to reduce the number of nodes. With this example data, however, this is not recommended as we would lose most nodes. As we are just interested in the presence of attacks and do not have any detailed information on the attacks, there is no relation variable to select. The method 'Count' may be selected for this example. Any other method will do as well.
4. You now have to choose two output files. The first file contains the network information. This file will be needed to visualize it. The second file only contains the information how many times each node appeared in the data. This information may be included to the network.

The results of this transformation are two tables. The first table (adjacency matrix) looks like a co-occurence matrix with the exception that it is not symmetrical. In the adjacency matrix, the senders are the rows of the table, whereas the objects are the columns. A number in row 3 and column 5 would therefore mean that the element on row 3 has a link to the element in column 5.

In the current example, you may see that Joe Biden does not utter any attack on other persons but that he is attacked by Paul Ryan once. Similarly, Paul Ryan is never the object of attacks but twice the subject.

Table 20: Adjacency Matrix

_Case	Biden, Joe	Christie, Chris	Cruz, Ted	Foreign Actor	Kerry, John	Obama, Barrack	Ryan, Paul	Tea Party	Van Hollen, Chris
Biden, Joe	0	0	0	0	0	0	0	0	0
Christie, Chris	0	0	0	1	0	0	0	0	0
Cruz, Ted	0	0	0	0	0	1	0	0	0
Foreign Actor	0	0	0	0	0	0	0	0	0
Kerry, John	0	0	0	1	0	0	0	0	0
Obama, Barrack	0	0	0	1	0	0	0	0	0
Ryan, Paul	1	0	0	0	0	1	0	0	0
Tea Party	0	0	0	0	0	0	0	0	0
Van Hollen, Chris	0	0	0	0	0	0	0	1	0

The second table has only two columns. The first contains the ID of the nodes (variable _Case) in the adjacency matrix table, the second column indicates the number of times a node appeared (Table 21).

Table 21: Attribute Matrix with node counts. Even nodes not appearing in the social network are listed here, such as the missing value

id	Count
	12
Biden, John	1
Christie, Chris	2
Cruz, Ted	3
Foreign Actor	3
Kerry, John	3
Obama, Barrack	6
Ryan, Paul	3
Tea Party	1
Van Hollen, Chris	4

14.14. Finding peak skin conductance during a movie

As the example dataset does not contain any time series data which may be used to detect peaks, an additional dataset with time series data from physiological measurements is used in this example. The dataset 'Example_Timeseries.txt' contains the heartrate, electrodermal resistance and blood pressure of a subject watching a short movie.

To find moments with increased or decreased conductance, you may use the function Peak Detection

1. Select the procedure 'Detect Peaks' from 'Time Series' and open the example time series data (Example_Timeseries.txt) (with header and tabstopps)
2. Select the input variables. The variable 'Time' contains the timestamps in seconds. The variable 'SkinCond' contains skin conductance data. Choose any name for the new variable which will be created (e.g.: Peak_Skin).

3. In this example we look for both positive and negative peaks. The confidence interval may be set to low 80% which means that a peak will only be considered a peak if the value is among the highest or lowest 10%.
4. Select a name and format for the output file and confirm (e.g.: Skincond.txt).

The result is a table exactly corresponding the input table with one additional variable. The variable has the value 0 for most of the time but contains few cases with value 1 (positive peaks) and -1 (negative peaks).

Since the data is quite extensive, R is used to visualize the result. As you may see from the picture, the function did identify regions with increased and reduced skin conductance but does not offer much information on the curve. Especially since small peaks which are detectable by eye are not indicated as they are within the 80% confidence interval.

The code in R was:

```
dta <- read.table("../Skincond.txt", header=TRUE, sep="\t")
plot (dta$Time, dta$SkinCond, type="l", main="Skin Conductance with
      detected peaks", xlab="Time in Seconds", ylab="Skin Conductance")
points(dta$Time[dta$Peak_Skin>0],
      dta$Peak_Skin[dta$Peak_Skin>0]*.2+2.3,col="green",pch="^")
points(dta$Time[dta$Peak_Skin<0],
      dta$Peak_Skin[dta$Peak_Skin<0]*.2+2.3,col="red",pch="v")
```

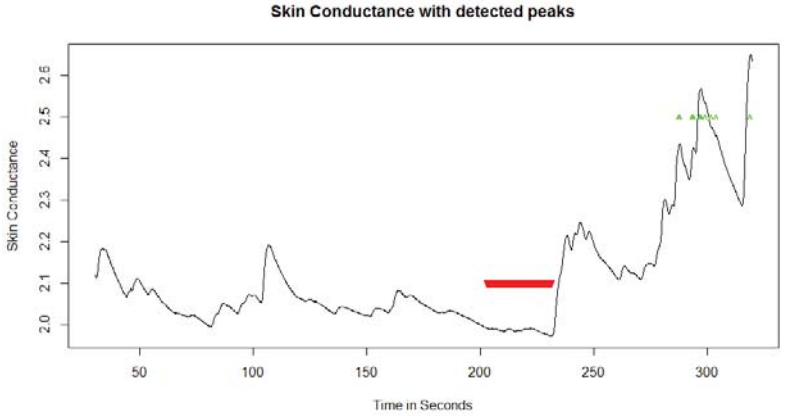


Figure 18: Skin conductance curve with indicated positive and negative peaks. Green triangles indicate positive peaks, red triangles indicate negative peaks.

14.15. Find peak increases in skin conductance

For this purpose, the same dataset as above is used, but the interest is slightly different. This time, the research interest does not concern the highest and lowest skin conductance over a long period of time, but small changes in skin conductance during this period. As the curve has been shown to exhibit a moving average, these changes may not be found by simply identifying high and low values. By subtracting the moving average from the curve, however, small changes may be observed easily.

1. Select the procedure 'Flatten moving average' from 'Time Series' and open the example time series data (Example_Timeseries.txt) (with header and tabstopps)
2. Select the input variables. The variable 'Time' contains the timestamps in seconds. The variable 'SkinCond' contains skin conductance data.
3. As Skin conductance takes some seconds to increase or decrease, it seems reasonable to set the gliding window for the moving average to 5-10 seconds.
4. Now you may enter two names for the two variables which are created (e.g.: Skin_MA and Skin_Dev)
5. Select a name and format for the output file and confirm (e.g.: Skincond.txt).

Again, the result is displayed in R as the data is too extensive and uninstructive as a table. To show the curves in one image and without overlap, the curves have been displaced vertically. The moving average (blue) is 0.1 points lower and the deviation of the moving average (red) is raised by 1.8 points in the graph. In the data, the moving average lies exactly on the time series data and the average of deviations is around 0.

The R code looks as follows:

```
dta <- read.table("../Skincond.txt", header=TRUE, sep="\t")
plot (dta$Time, dta$SkinCond, type="l", ylim=c(1.7,2.7), main="Skin
      Conductance with moving average and deviation", xlab="Time in
      Seconds", ylab="Skin Conductance", lwd=2)
lines(dta$Time, dta$Skin_MA-.1,col="blue")
lines(dta$Time, dta$Skin_Dev+1.8,col="red",pch="v")
```

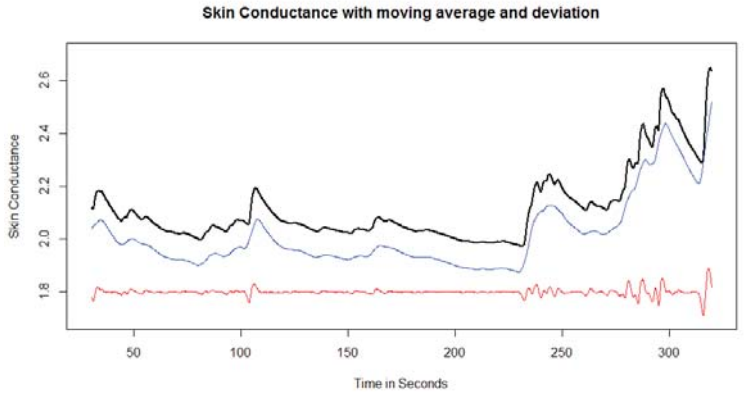


Figure 19: Graphical illustration of the moving average (blue) and deviation (red) of the skin conductance curve. The moving average was computed using a 5 second gliding window.

In a next step, the peaks of the deviation curve (now stripped from the moving average) may be used to detect the real peaks of the curve. As the curve only has positive peaks with intervals of low relaxation, only positive peaks are considered now.

6. Select the procedure 'Detect Peaks' from 'Time Series' and open the file created above (e.g.: Skincond.txt)
7. Select 'Time' and 'Skin_Dev' as input variables and an output variable (e.g.: Skin_Peaks).
8. Select positive peaks and a 90% confidence interval to consider only the top 5% of values.
9. Select a name and format for the output file and confirm (e.g.: Skincond.txt).

Again, the result is visualized in R. The code is as follows:

```
dta <- read.table("../Skincond.txt", header=TRUE, sep="\t")
plot (dta$Time, dta$SkinCond, type="l", main="Skin Conductance with real
      peaks", xlab="Time in Seconds", ylab="Skin Conductance")
points(dta$Time[dta$Skin_Peaks>0], dta$Skin_Peak[dta$Skin_Peaks>0]*.2+2.3,
      col="darkgreen",pch="^",cex=2)
```

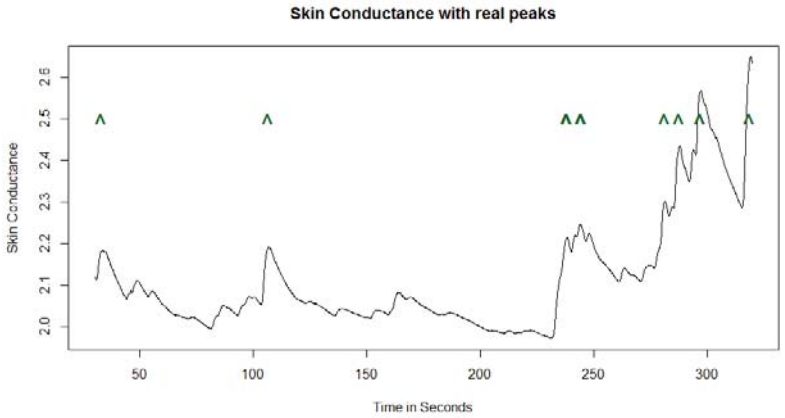


Figure 20: Detected peaks of the skin conductance curve when stripped of the moving average. It may be observerd that not all peaks have been found due to the high confidence interval of 90% which was selected.

In this example you may see that the routine detected large and small peaks independent from their baseline skin conductance. Whenever there is a peak increase, a peak is detected. The new variable may now be used to identify the exact time when sudden increase of skin conductance occurs.

Anhang B3: Aeglos

Online-Ressource:

Im Folgenden wird die Version bei Einreichung dieser Dissertation dargestellt. Eine vollständige und ständig aktualisierte Version der Dokumentation und Anleitung, inklusive Beispielprogramme, sind Online verfügbar unter:

http://www.ipmz.uzh.ch/de/Abteilungen/Medienpsychologie/Recource

Aeglos 0.2 (alpha)

Quick tutorial

Martin Wettstein

Aeglos, n [aeglos] 1. Angrist extension for semi-automated content analysis. The main function of Aeglos is the training and application of naive Bayes classification for categories in content analyses. In addition to this core function, the program also performs fully automated analyses of texts.
2. (from Sindarin: snow point): Name of the spear of Gil-Galad with which Sauron was wounded in Dagor Dagorath to enable Isildur to cut off his finger.

This program is still in development and in the stage of alpha-testing.

The same applies to this documentation. There may be errors and inconsistencies which are yet to be resolved.

Zürich, July 2014

Contents

Chapter 1. Introduction **3**
1.1. Key terms in this documentation 3
1.2. Language Settings 4

Chapter 2. Naive Bayes Classification in Aeglos **5**
2.1. Optimizing Bayes classification for large texts 5

Chapter 3. Functions of the Aeglos GUI **8**
3.2. Create Corpus from Codesheets and Textfiles 9
3.3. Create Corpus from a directory of textfiles 9
3.4. Train Aeglos 10
3.5. Test Aeglos: Detailed Test with one Text 11
3.6. Test Aeglos: Reliability Test with Test Corpus 11
3.7. Test Aeglos: Batch testing with different Settings 12
3.8. Automatically predict categories: Predict Values 12
3.9. Automatically predict categories: Predict Probabilities 13
3.10. Tool: Create Term-Document-Matrix 14
3.11. Tool: Co-occurrence frequency with keyword 14
3.12. Tool: Split collocated textfiles 15

Chapter 1. Introduction

Aeglos (Angrist Extension Generating Listselections for Optimal Speed) is a python script which may be used as a GUI to prepare, train, and apply automated content analyisis or as a module to be included to Angrist for semi-automated content analysis.

Aeglos uses a form of naive Bayes classification, which was adapted to enable the reliable classification of long texts, to predict the most probable value for any given category in a content analysis. This function may be used to automatically code categories or to predict the most probable choice of human coders in manual content analyses and offering this choice as a default value during data entry. The script was designed to work perfectly well with the Angrist coder interface (Wettstein 2014) and may easily be applied to quantitative content analyses using this tool. In addition to the core functions of Aeglos, the program is also able to perform automated analyses of word co-occurrences.

In this quick reference, the usage of Aeglos is explained in a step-by-step manual. In chapter 2, we begin by explaining the logic behind Bayes classification and the formulae behind the calculations. Then, the central functions of the Aeglos GUI are explained. In chapter 4, the inclusion of Aeglos in Angrist is explained in detail. Finally, in chapter 5, technical data and examples are shown.

1.1. Key terms in this documentation

In this documentation, some terms are used which have different meanings in linguistics, computational linguistics, communication science, and political science. It is therefore necessary to define the use of these words in this documentation.

In quantitative content analyses, texts are categorized manually or automatically in a previously defined set of text categories. Each category may have different values which are determined for each text and stored in a codesheet, which is a table containing one variable for each category. All categories and values are defined in a codebook which holds detailed information and a coding scheme for each category, as well as some examples. As an example you may consider the following set of categories:

Variable	Category	Values
ID	Identifier of the text	running number
RELEVANCE	Relevance of the text for this analysis	1: Relevant 0: Irrelevant
REF_GEO	Geographic reference of this text	1: Germany 2: United Kingdom 3: Other
ISSUE	Broad issue of the text	1: Politics 2: Economy 3: Sports

In a content analysis of four texts using this set of categories, a codesheet is produced which holds one variable for each category. The cases of the codesheet are the texts, the values for each variables are the values of the categories for this text.

ID	RELEVANCE	REF_GEO	ISSUE
10	1	1	2
11	0	1	1
12	1	1	3
13	0	2	3

In order to work with Aeglos, corpora are required. A corpus in Aeglos is a codesheet with one additional variable containing the text that was classified. A corpus would therefore look like that:

ID	RELEVANCE	REF_GEO	ISSUE	Fulltext
10	1	1	2	Steigende Mieten Hendricks will Wohnungsbau fördern. // Bundesbauministerin Barbara Hendrix will das Problem steigender Mieten jetzt an der Wurzel bekämpfen. Sie plant ...
11	1	1	1	Deutsch-amerikanische Spionageaffäre: Schäuble entsetzt über "so viel Dummheit" // Die Bundesregierung zeigt sich empört über den möglichen Spionagefall im Verteidigungsministerium...
12	0	1	3	Deutschlands Finalgegner Argentinien: Ihr schon wieder! // Argentinien bezwingt die Niederlande 4:2 im Elfmeterschießen und steht im WM-Finale. Jetzt geht es gegen Deutschland - ...
13	0	2	3	Alex Song konnte sich seit seinem Wechsel vom FC Arsenal zum FC Barcelona nicht durchsetzen. Nun jagt ihn die halbe Premier League. Auch der AS Monaco will ihn haben. ...

These corpora are usually stored in tabulator spaced text files and may be used for all functions of Aeglos, such as training, automated prediction of categories, the generation of term-document matrices, and term mapping.

1.2. Language Settings

In most routines, the texts are preprocessed to temove suffixes from words and reduce them to their word stems. For this stemming procedure, it is important to select the correct language when processing texts.

Up to now, stemming routines for the follwing languages have been implemented in Aeglos:

- English
- German
- French
- Italian
- Swedish
- Dutch

All stemming algorithms were implemented according to the documentation of the snowball project (http://snowball.tartarus.org/algorithms/). An addition of further languages is possible and will be done in time. If the language of texts in an analysis is not yet implemented, you should choose 'other language' when asked for the language of texts. In these cases, no stemming will be applied.

Chapter 2. Naive Bayes Classification in Aeglos

In Aeglos, a naive Bayes Classifier (NBC, see Manning & Schütze 2001) is used to predict the probability of certain values of categories. NBC is a trained statistical procedure to classify unknown texts by using information from known and correctly classified texts.

The central idea behind the NBC algorithm is the Bayes theorem of conditional probability. It holds that the conditional probability of A, given B (P(A|B)) is equal to the product of the probability of A and the conditional probability of B, given A, divided by the probability of B (formula 1).

$$P_{(A|B)} = \frac{P_{(A)} P_{(B|A)}}{P_{(B)}} \quad (1)$$

In text classification, this formula may be used to calculate the conditional probability of a certain value of a category, given the words in the text. The central assumption of NBC algorithms is therefore that the probability of a given value *k* from a set of values possible for category *K* is proportional to the product of conditional probabilities of each word *w* in the text T, given *k*, multiplied by the probability of the occurrence of *k* (Formula 2).

$$P_{(k|T)} \propto P_{(k)} \prod_w^T P_{(w|k)} \quad (2)$$

When the conditional probabilities of all values k in K are calculated using this formula, the value reaching the highest score is most probably the correct value for this category in this given text.

In order to compute P(k|T) for any text, all that is required is the overall probability of k (P(k)) and a table of all conditional probabilities of all words, given value k. This table may be prepared in advance and used for each classification. In NBC, the training of the algorithm consists in calculating the probability tables for all values and as many words as possible. To do so, a set of previously classified texts may be used.

2.1. Optimizing Bayes classification for large texts

NBC algorithms are used in spam filters to judge the relevance of emails from its content and in machine translation to choose the most appropriate translation from the context of a word in ambiguous situations. In these cases, however, only small amounts of text are used in the algorithm. In content analyses of large and complex texts, such as newspaper articles, the predictive quality of the formula is hapered by several problems. First, the low global occurrence of some of the words results in very low values for *P(w|k)*. Second, the computation of products of several hundred of factors which are close to zero may result in long floating point numbers. Third, there are many stopwords and irrelevant words which do not contribute to the prediction of the category but require computational power. Fourth, expecially in languages with few composita, single words may not be discriminant. For theses languages, short sequences of words should also be taken into account. Finally, there may be very relevant but rare words which are very important for the prediction of the correct value and should be weighed more.

In order to overcome these problems, the formula for the NBC is slightly altered in order to work well with large texts. In order to reduce the problem of very low conditional probabilities in the formula, the quotient of the original Bayes Theorem is included in the formula. Thereby, the overall probability of each word is taken into account and the factors are close to 1 in most cases:

$$P_{(k|W)} \propto P_{(k)} \prod_w^T \frac{P_{(w|k)}}{P_{(w)}} \quad (3)$$

Even with this modification, the formula may result in scores for the different text categories which are several decimal points apart from each other. It is possible to choose the highest score but there is no reasonable way to determine the distance between the highest and the next highest score which would be required to guess the certainty of the program. For this reason, Aeglos uses the T-root of this score which results in the harmonic means of all conditional probabilities of category k, given text T.

$$P_{(k|W)} \propto \sqrt[|T|]{P_k \cdot \prod_w^T \frac{P_{(w|k)}}{P_{(w)}}} \quad (4)$$

Using products and large roots, this formula is expensive in computation and may result in large floating point numbers. When computing the probabilities of text categories for a set of large texts, it would be better to have fast and cheap operations. It is therefore suggested to use the logarithm of this function:

$$\ln\left(P_{(k|T)}\right) \propto \frac{v \cdot \ln\left(P_{(k)}\right) + \sum_w^T \ln\left(\frac{P_{(w|k)}}{P_{(w)}}\right)}{|T|} \quad (5)$$

In order to use this formula at the lowes possible cost, the training should not consist in just calculating the conditional probabilities P(w|k) but in calculating the logarithms used as summands in this formula for each word and category. By storing these values in probability tables, the algorithm just has to compute the sum of a series of stored values. Another advantage of this formula is the avoidance of numbers close to 0.

As some languages (such as English) use few composita, not only words but short sequences of words should be taken into account as well when analyzing a text. In computational linguistics, these short sequences are called n-grams. Usually, bigrams (two consecutive words) or trigrams (three consecutive words) should be enough to capture fixed terms which are expressed by a sequence of words such as "secretary of state", "next year", "voted against" which are meaningful in this sequence but would lose their significance if cut up in single words.

In order to tackle the different significance of single words, their discriminative power in the analysis has to be assessed. If the values which are stored in the probability tables are the logarithms of the conditional probabilities divided by the overall probabilities of each word, the values for different categories for each word are centered around 0. Words with a low discriminative power, which occur similarly in different categories have values close to 0 as the conditional probabilities are near to the overall probability of each word. Words which occur more often in texts of one category than in others, have higher values as their conditional probabilities differ from the overall probability. The discriminative power of each word may therefore be judged from the maximal value over all categories:

$$D_w = \max_{k\ in\ K}\left(abs\left(\ln\left(\frac{P_{(w|k)}}{P_{(w)}}\right)\right)\right) \quad (6)$$

In order to remove irrelevant words from the calculation, words with a low discrimination may be excluded. Furthermore, it is recommended to weigh the conditional probability with the discrimination to overweigh words with a high discrimination and give low weight to words with a lower discrimination:

$$\ln\left(P_{(k|T)}\right) \propto v \cdot \ln\left(P_{(k)}\right) + \frac{\sum_w^T \ln\left(\frac{P_{(w|k)}}{P_{(w)}}\right) + D_w}{|T|} \quad (7)$$

Transforming formula 7 by applying the exponential function, the score achieved is again relational to the actual conditional probability which is looked for:

$$P_{(k|T)} \propto e^{v \cdot \ln(P_{(k)}) + \frac{\sum_w^T \ln\left(\frac{P_{(w|k)}}{P_{(w)}}\right) + D_w}{|T|}} \quad (8)$$

Applying formula 8 to all text categories possible, given an unknown text, a score relational to the probability of each category for this text may be gathered. By scaling these scores to a total sum of 1, a pseudo-probability for each category may be calculated and used for further calculations.

Aeglos uses formula 8 to calculate the score for each category and returns the most probable category with its pseudo-probability as an output. This output may then be used to reach a coding decision. The minimal discrimination needed for any word to be included may be defined by the user and may depend on the distribution of categories.

Chapter 3. Functions of the Aeglos GUI

3.1. Get Universe of n-grams

For computational reasons, it is not useful to compute the conditional probability for each possible n-gram in a corpus, given each text category. This would result in very large probability tables, most lines only containing 0% or 100% as the word occurred only once, either in a text with a given text category or without it. It is therefore recommended to narrow down the universe of possible n-grams to words and n-grams actually occurring in more than 1% of the texts and in less than 99% of the texts. This will remove all very rare and all ubiquious terms from the list of considered n-grams.

In order to do so, the function 'get universe of n-grams' uses all texts in a given folder (probably including sub-directories) to create a list of words which may actually be taken into account in the training of Aeglos.

Options:

Input Directory

You may select any directory containing textfiles on your computer. All textfiles will be rendered to n-grams in order to provide a list of all n-grams encountered in real texts. As this function will only have to be used once for each language, you may as well use a very large folder of textfiles and give the program some minutes to complete its task.

Include Subdirectories

In some cases, your texts may be organized in sub-directories. If you select 'yes', all subdirectories are taken into account. If you select 'no' only the chosen directory is used to find n-grams.

Language of texts

Select the language in which the texts are written. The function will use a lemmatizer to extract word stems from the words in the texts. This step in the function is strongly dependent on the language used.

Length of n-grams

Choose the maximum number of consecutive words you wish to extract from the texts. If you select 3, all words, bigrams and trigrams will be extracted. Please consider that selecting a higher number of consecutive words results in longer computation times. In most cases, trigrams are the maximum required to reach a satisfacory reliability of Aeglos.

Lowest (highest) word frequency?

You may choose up to which frequency of words a word should be discarded from the list. As words occurring in more than 99% of the texts are highly unlikely to occur more in texts of one category than another one, it is safe to discard all words occurring in less than 1% and more than 99% of the texts. You may, however, also choose to exclude words which occur in less than 5% or 10% or in more than 95% or 90%, respectively.

Output File

Choose any file in which the complete list should be written in the end. The file will contain a table with two variables. The first variable 'Ngram' contains the words and consecutive words. The second variable 'Count' contains the number of texts the n-gram was encountered in.

The output table is sorted alphabetically.

3.2. Create Corpus from Codesheets and Textfiles

In order to train and use Aeglos, a manually annotated corpus needs to be present in which the value of a category in a collection of texts is known. This training corpus ideally contains more than 100 texts with abundant examples for all values possible for each cateogry. Likewise, you may want to create test corpora or corpora of unknown texts which are to be predicted and used in reliability tests.

Corpora in Aeglos are tables which contain at least one column with values for one text category and one column with the complete text to which the text categories apply. There is no limit to the number of variables in the file and to the number of categories to be worked with. Usually, the table is stored in a textfile with a separator which does not appear in any of the texts in the text column and contains the names of the variables in the first row.

If only the codings for each text are available but there is not yet any column holding the correpsonding texts, the functzion 'Create corpus' may be used to combine textfiles with the table of codings. The textfiles should be in ANSI format (ASCII characters) and collected in a folder on your computer. One column of the codesheet must contain the filename of the text which was classified.

Note that the size of a corpus is almost equal to the size of all textfiles used in this corpus, as they are stored completely in the new variable. If you work with more than 10'000 texts it is advised to create several smaller corpora instead of one large corpus which may reach sizes over 30MB.

Options:

Codesheets
The first input of this function is a table of codings for the texts. This table has to include one column identifying the name of the textfile associated with each line. This identifier will be used to combine the contents of a textfile with each line.

Textfiles
Here, you may specify the folder in which the textfiles are collected. The folder should contain all files specified in the codesheet. Any line containing a missing file will be removed from the corpus.

Name of Text variable
Here, you may specify a name for the newly created column containing all texts (e.g. 'Fulltext')

Name of Text ID
From all the columns of the codesheet you may select the column which contains the identifiers (names of the textfiles).

Output File
Enter a name for the corpus to be created. The table will be stored as a textfile with header and a separator.

3.3. Create Corpus from a directory of textfiles

In the absence of existing codesheets, a corpus may also be created from a directory of textfiles. The corpus will then only contain two variables, one denoting the filename of the text, the other holding the complete text of each file. These corpora may be used in automated classification of texts by Bayes classification or for the generation of term-document matrices.

Note that the size of a corpus is almost equal to the size of all textfiles used in this corpus, as they are stored completely in the new variable. If you work with more than 10'000 texts it is advised to create several smaller corpora instead of one large corpus which may reach sizes over 30MB.

Options:

Textfiles
Here, you may specify the folder in which the textfiles are collected. Only files with the extension '.txt' are used in the generation of a corpus.

Name of Text variable
Here, you may specify a name for the newly created column containing all texts (e.g. 'Fulltext')

Name of Filenames Variable
Here, you may specify a name for a variable holding the filename of each text.

Output File
Enter a name for the corpus to be created. The table will be stored as a textfile with header and a separator.

3.4. Train Aeglos

When a corpus is prepared, you may train the NBC for any category present in the codesheet. In training, a table of conditional probabilities is created for each value and word which may then be used to classify new texts. Depending on the number of texts in your training corpus and the lenght of these texts the training may take several minutes.

Train Input
Choose the training corpus you wish to use for the training. A corpus has to contain at least one variable to train and one column which holds the texts associated with each coding.

Complete Wordlist
Here, you may specify the universe of n-grams you wish to use in this analysis. All words and n-grams which are not present on this list are excluded from training. All words which are on the wordlist but do not occur in the training corpus are excluded automatically.

Wordlist Variable
Specify the variable which holds the actual words of the wordlist.

Fulltext Variable
Specify the vvariable in the corpus which hods the texts associated with each coding.

Language
As Aeglos uses word stems instead of full words, you have to specify the language of the texts. The rules for stemming words are strongly language dependent.

Length of n-grams
You may specify the maximal number of consecutive words you wish to include in the training. Consider that it does not make any sense to use a higher number than present in the wordlist. As all terms which are not present in the wordlist are ignored, all n-grams longer than the ones in the wordlist are excluded.

Training Variables

You may select any number of variables to be trained by Aeglos. For the training, the conditional probabilities for each term in the wordlist and each value of the variables is computed and written to a probabily table to be used by Bayes classification.

Output Table

You may specify the name of the probability table holding the results of the training. These training tables or probability tables may be used in the automated prediction of categories or in the test of prediction accuracy.

3.5. Test Aeglos: Detailed Test with one Text

The most straightforward test of Aeglos is a detailed analysis of one text. Aeglos requires a probability table from a previous training and an unknown text to perform this analysis.

(Details of the detailed test are not yet implemented. At the moment it just does a test and returns the probabilities for the values)

TO BE DETERMINED

3.6. Test Aeglos: Reliability Test with Test Corpus

In order to test the reliability of a trained corpus you may test the predictions in a test corpus which contains the previously correctly coded categories. This function is also recommended as a test of the predictability of a trained cateogry. If you test the training on the training corpus itself, the result should be very reliable if it is possible to train a category. If the training corpus itself may not be predicted reliably you may have a category which is not predictable by NBC.

As a result, this function displays the reliability scores on screen and writes a new table containing the predictions made in the test for further investigation of errors.

Measures for reliability:

- Precent Agreement: The percent of texts for which the prediction of Aeglos is equal to the manual coding.
- Precision (Only for dichotomous (0/1) variables): The share of texts, coded as 0, in the texts predicted to have the value 1.
- Recall (Only for dichotomous (0/1) variables): The share of texts coded as 1, which were predicted to be 0.
- Cohen's Kappa: Percent agreement, corrected for per chance agreement.
- Mean Probability: The mean probability with which Aeglos predicted the values of the test corpus.

Test Corpus

First, a test corpus has to be specified. This corpus has to contain the variable to be classified by Aeglos. A test is only possible if this variable is named exactly the same as the variable which was previously trained in Aeglos. The routine will compare the values of the stored variable with automated classifications.

Probability Table

To test the predictions of Aeglos, a probability table from a previous training has to be specified. You may only use the probability tables written by Aeglos in this routine.

Language

Here, you may specify the language which is used in the corpus. The language is important for the stemming algorithm of Aeglos. Using wrong language settings may impair the quality of predictions.

Length of n-Grams

Here, you may specify the number of consecutive words to be counted as a single term. If you choose a higher number than the one used in the training, the maximal length of n-grams in the probability table will be used.

Minimum discrimination

To enhance the performance of Aeglos for long texts you may choose a minimum discrimination for terms to be used in the prediction of categories. Choosing a value of 0.3 is recommended for most cases as it reliably removes stop words from the list of relevant words. This increases both the quality and the precision of the prediciton.

Weight of discrimination

You may choose to weigh the contribution of each term to the overall prediction by its maximum discrimination. This option will assign more importance to terms which have been found to discriminate strongly between categories.

Variable to be analyzed

Here, you may select the variables to be tested. Only variables for which probabilities are found in the probability table and which are present in the corpus may be tested. If no variable is eligible, the corpus and training table are not compatible as they do not share a variable.

You may add any number of variables to the list to be analyzed.

Text variable

From all variables within the corpus and the training table you may select the one containing the full text of each text in the corpus.

Output Table

You may specify the file to which the corpus containing the automated predictions should be written for later use or analysis. The report of the reliability test will be shown in the text output in Aeglos.

3.7. Test Aeglos: Batch testing with different Settings

For the reliability test you may choose some setting, such as the minimum discrimination of a word required to be included in the analysis and whether the probabilities are to be weighed by the discrimination. As these settings may change the results of the reliablity test it is recommended to test different settings.

Using this function you may perform a reliability test with different combinations of settings without manually restarting Aeglos after each test. This saves time and returns a systematic overviev of results in the end.

The options for this function are analogous to the options described in the reliability test above. Hovewer, you may choose several parameters for minimum discrimination ans weight of discrimination to be tested simultaneously.

3.8. Automatically predict categories: Predict Values

Aeglos may be used to predict the values of categories in coproa of unknown texts. You may predict the values of any trained category. Aeglos returns a table containing the predicted values and the probability with which this prediction is correct.

Corpus to be predicted
First, a test corpus has to be specified which holds the texts to be predicted.

Probability Table
To predict categories using Aeglos, a probability table from a previous training has to be specified. You may only use the probability tables written by Aeglos in this routine.

Language
Here, you may specify the language which is used in the corpus. The language is important for the stemming algorithm of Aeglos. Using wrong language settings may impair the quality of predictions.

Length of n-Grams
Here, you may specify the number of consecutive words to be counted as a single term. If you choose a higher number than the one used in the training, the maximal length of n-grams in the probability table will be used.

Minimum discrimination
To enhance the performance of Aeglos for long texts you may choose a minimum discrimination for terms to be used in the prediction of categories. Choosing a value of 0.3 is recommended for most cases as it reliably removes stop words from the list of relevant words. This increases both the quality and the precision of the prediciton.

Weight of discrimination
You may choose to weigh the contribution of each term to the overall prediction by its maximum discrimination. This option will assign more importance to terms which have been found to discriminate strongly between categories.

Variable to be analyzed
Here, you may select the variables to be tested. Only variables for which probabilities are found in the probability table may be predicted. You may add any number of variables to the list to be analyzed.

Text variable
From all variables within the corpus and the training table you may select the one containing the full text of each text in the corpus.

Output Table
You may specify the file to which the corpus containing the automated predictions should be written.

3.9. Automatically predict categories: Predict Probabilities

Especially for further processing of predictions it is useful to predict the probabilities of all possible values of a category instead of only the most probable choice. The data may then be used to rate texts based on their probability of a certain value.

The prediction of value probabilities creates one additional variable for each value of the category. In these variables, the probabilities for each value is written.

Corpus to be predicted
First, a test corpus has to be specified which holds the texts to be predicted.

Probability Table
To predict categories using Aeglos, a probability table from a previous training has to be specified. You may only use the probability tables written by Aeglos in this routine.

Language
Here, you may specify the language which is used in the corpus. The language is important for the stemming algorithm of Aeglos. Using wrong language settings may impair the quality of predictions.

Length of n-Grams
Here, you may specify the number of consecutive words to be counted as a single term. If you choose a higher number than the one used in the training, the maximal length of n-grams in the probability table will be used.

Minimum discrimination
To enhance the performance of Aeglos for long texts you may choose a minimum discrimination for terms to be used in the prediction of categories. Choosing a value of 0.3 is recommended for most cases as it reliably removes stop words from the list of relevant words. This increases both the quality and the precision of the prediciton.

Weight of discrimination
You may choose to weigh the contribution of each term to the overall prediction by its maximum discrimination. This option will assign more importance to terms which have been found to discriminate strongly between categories.

Variable to be analyzed
Here, you may select the variables to be tested. Only variables for which probabilities are found in the probability table may be predicted. You may add any number of variables to the list to be analyzed.

Text variable
From all variables within the corpus and the training table you may select the one containing the full text of each text in the corpus.

Output Table
You may specify the file to which the corpus containing the automated predictions should be written.

3.10. Tool: Create Term-Document-Matrix

This tool creates a term document matrix (TDM) from a corpus, optionally using a pre-existing list of words to do so. In order to create the matrix, a document identifier has to be in the corpus (several cases of the corpus may belong to the same document) and a column holding all texts.

The output is a matrix of documents X terms with count data. The matrix may then be used for further processing.

3.11. Tool: Co-occurrence frequency with keyword

This tool calculates the frequency of co-occurrence of any given number of words with a keyword. As an input, the tool requires a corpus and a list of words, preferrably a universe created by Aeglos.

The tool generates a list containing the probabilities for any word from the list to be in a text when the keyword is mentioned (PW2_W1) and the probability of the keyword being mentioned in a text when any given word is inside (PW1_W2). In addition, a co-occurrence score is computed which is calculated as the product of these two probabilities.

3.12. Tool: Split collocated textfiles

This tool reads a texfile containing multiple texts to split it into a series of enumerated textfiles, each containing one text. This tool may be used to separate the texts received from archives, such as LexisNexis or Factiva.

Anhang C: Akademischer Lebenslauf

Kontakt:
Martin Wettstein
Brunnenwiesenstrasse 62
8212 Neuhausen
m.wettstein@ipmz.uzh.ch

Tel. Mobil: 079 567 84 77
Tel. Büro: 052 672 10 45

Persönliches:
Geboren: 06.02.1980
Nationalität: CH
Kinder: Timo (30.11.2010), Elanor (11.04.2013)

Ausbildung

seit 2009 ..... Doktoratsstudium an der Philosophischen Fakultät der Universität Zürich

2009-2013 ..... NCCR Doctoral School

2004-2009 ..... Lizentiat der Medien- und Kommunikationswissenschaft, Soziologie und Computerlinguistik, Universität Zürich

2000-2003 ..... Studium der Biochemie, ETH Zürich

1993-2000 ..... Eidgenössische Matura, Kollegium St.Fidelis, Stans

Berufliche Erfahrung

seit 2009 ..... Wissenschaftlicher Mitarbeiter am IPMZ, Abteilung „Media Psychology & Effects“, Prof. Dr. W. Wirth.

Sonstige Aktivitäten

seit 2014 ..... Sprecher der Fachgruppe Methoden der SGKM

Academic Record

Publikationen in Fachzeitschriften:

Wettstein, M. (In Press). Quantitative Ursachenbestimmung medialer Aufmerksamkeitsschübe. *Publizistik, 60*(3).

Wettstein, M. (2012). Frame Adoption in Referendum Campaigns: The Effect of News Coverage on the Public Salience of Issue Interpretations. *American Behavioral Scientist, 56*(3), 318–333. doi:10.1177/0002764211426328

Wettstein, M. (2010). Politische Partizipation über Soziale Netzwerkdienste: Qualitative und quantitative Charakterisierung der Facebook-Gruppe als Mittel zur politischen Meinungsäusserung und Partizipation. *Medien Journal, 34*(3), 4–21.

Wirth, W., Matthes, J., Schemer, C., Wettstein, M., Friemel, T., Hänggli, R., & Siegert, G. (2010). Agenda Building and Setting in a Referendum Campaign: Investigating the Flow of Arguments Among Campaigners, the Media, and the Public. *Journalism & Mass Communication Quarterly, 87*(2), 328–345.

Beiträge in Sammelbänden:

Wettstein, M., Wirth, W., & Reichel, K. (2015). Zum Problem der Mehrfachcodierung: Sind drei wirklich genug?: Eine systematische Fehleranalyse. In W. Wirth, K. Sommer, M. Wettstein, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 12. Qualitätskriterien in der Inhaltsanalyse* (pp. 219–236). Köln: Herbert von Halem Verlag.

Wirth, W., Wettstein, M., Kühne, R., & Reichel, K. (2015). Theorie und Empirie des Codierens: Personelle und situative Einflussfaktoren auf Qualität und Quantität des Codierens bei der Inhaltsanalyse. In W. Wirth, K. Sommer, M. Wettstein, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 12. Qualitätskriterien in der Inhaltsanalyse* (pp. 96–118). Köln: Herbert von Halem Verlag.

Wettstein, M. (2014). "Best of Both Worlds": Die halbautomatische Inhaltsanalyse. In K. Sommer, M. Wettstein, W. Wirth, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse* (pp. 16–39). Köln: von Halem.

Sommer, K., Wettstein, M., Wirth, W., & Matthes, J. (2014). Zum Schattendasein der automatisierten Inhaltsanalyse: Ein Vorwort. In K. Sommer, M. Wettstein, W. Wirth, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse* (pp. 9–15). Köln: von Halem.

Wirth, W., Wettstein, M., Reichel, K., & Kühne, R. (2013). Äquivalenzprüfung als Standard in international vergleichenden Inhaltsanalysen. In T. Naab, D. Schlütz, W. Möhring, & J. Matthes (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 9. Standardisierung und Flexibilisierung als Herausforderungen der kommunikations- und publizistikwissenschaftlichen Forschung* (pp. 258–284). Köln: Herbert von Halem Verlag.

Wettstein, M. (2012). Term-Mapping zur komparativen Analyse öffentlicher Debatten: Eine Anwendung der Smallest Space Analysis für Inhaltsanalysen. In B. Stark, M. Magin, O. Jandura, & M. Maurer (Eds.), *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 8. Methodische Herausforderungen Komparativer Forschungsansätze* (pp. 138–157). Köln: Herbert von Halem Verlag.

Wettstein, M. (2012). Politische Partizipation im Social Web: Hinweise zum Aufenthaltsort des totgesagten politischen Engagements jugendlicher Internetnutzer. In U. Dittler (Ed.), *Aufwachsen in sozialen Netzwerken. Chancen und Gefahren von Netzgemeinschaften aus medienpsychologischer und medienpädagogischer Sicht* (pp. 129–146). München: kopaed.

Herausgeberschaften:

Sommer, K., Wettstein, M., Wirth, W., & Matthes, J. (Eds.). (2014). Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 11. Automatisierung in der Inhaltsanalyse. Köln: von Halem.

Wirth, W., Sommer, K., Wettstein, M., & Matthes, J. (Eds.). (2015). *Methoden und Forschungslogik der Kommunikationswissenschaft: Vol. 12*. *Qualitätskriterien in der Inhaltsanalyse* (neue Ausg). Köln: Herbert von Halem Verlag.

Sonstige Publikationen:

Wettstein, M. (2014). *Angrist 1.2: Documentation and Reference for the Coder Interface*. Retrieved from http://www.ipmz.uzh.ch/Abteilungen/Medienpsychologie/Recource/Angrist/ANGRIST_1-2-en.pdf

Präsentationen:

Wirz, D. S., Wettstein, M., Schulz, A., Müller, P., Schemer, C., & Wirth, W. (2015, Januar). *Die unbeabsichtigte Komplizenschaft von Populisten und Boulevardmedien: Wirkung populistischer Appelle auf Zeitungsleser.* Vortrag auf der Jahrestagung der Fachgruppe für Rezeptions- und Wirkungsforschung der Deutschen Gesellschaft für Publizistik- und Kommunikationswissenschaft (DGPuK), Bamberg.

Wettstein, M. (2014, October). Agentenbasierte Simulation von Messintervallen: Auf den Spuren des Quasi-Statistischen Organs. FG Methoden der DGPuK, München, D.

Schulz, A., Wettstein, M. & Wirth, W. (2015, May). *Der Induktionsschluss beim Publikum: Empirische Evidenz für die Extrapolationshypothese im persuasive press inference Modell.* Vortrag auf der Jahrestagung der Deutschen Gesellschaft für Publizistik- und Kommunikationswissenschaft (DGPuK), Darmstadt.

Schulz, A., Wettstein, M. & Wirth, W. (2014, Mai). *All Hostile Media. Consonance of News Reporting as Moderator of the Hostile Media Effect.* Vortrag auf der Jahrestagung der International Communication Association (ICA), Seattle/WA.

Wettstein, M., & Wirth, W. (2014, May). *The Impact of Uncivil and Insincere Comments on the Deliberative Quality of Online Discussions*. International Communication Association; 2014 Annual Meeting, Seattle, WA.

Wirz, D., Ernst, N., Büchel, F., Schulz, A., Wettstein, M., Engesser, S., Schemer, C., Esser, F. & Wirth, W. (2014, Mai). *Populism and the Media Forming an Unholy Alliance: An Integrative Framework*. Vortrag auf der Jahrestagung der International Communication Association (ICA), Seattle/WA.

Wettstein, M. (2014, April). *Computerunterstützte Analyse öffentlicher Debatten: Methodeninventar für eine induktive und effiziente quantitative Inhaltsanalyse*. Jahrestagung der SGKM - Methodenpanel. Schweizerische Gesellschaft für Kommunikations- und Medienwissenschaften (SGKM), Zürich.

Schulz, A., Dingerkus, F., Wettstein, M., & Wirth, W. (2014, January). *Konsonanz und Hostile Media Effekt: Eine experimentelle Untersuchung der Wirkung von Konsonanz auf feindliche Medienrezeption am Beispiel des Konflikts zwischen Schulmedizinern und Homöopathen*. Vortrag auf der Jahrestagung der Fachgruppe für Rezeptions- und Wirkungsforschung der Deutschen Gesellschaft für Publizistik- und Kommunikationswissenschaft (DGPuK), Hannover.

Kühne, R., Schemer, C., Wettstein, M., Reichel, K., & Wirth, W. (2013, June). Assessing the Quality of Media Debates on Unemployment in Six European Countries. Paper presented at the Annual Conference of the International Communication Association (ICA), June 17-21 2012, London.

Wettstein, M. (2013, June). *Finding patterns of co-occurrence: Explorative Hierarchical Cluster Analysis with Automated Item Exclusion.* International Communication Association. Annual Convention of the International Communication Association 2013, London, London.

Russi, L., Siegert, G., Krebs, I., & Wettstein, M. (2012, November). *Wettbewerb und publizistische Vielfalt: Eine theoretische Modelierung und Fuzzy Set Analyse europäischer Zeitungsmärkte*. Jahrestagung der Fachgruppe Medienökonomie der DGPuK, Dortmund.

Wettstein, M. (2012, September). *"Best of both Worlds": Die halbautomatische Inhaltsanalyse*. Jahrestagung der Fachgruppe Methoden der DGPuK, Zürich.

Wirth, W., Wettstein, M., Reichel, K., & Kühne, R. (2012, September). *Zur "Black Box" des Codierprozesses: Lehren aus der Beobachtung von Codierern in ihrer natürlichen Umgebung*. Jahrestagung der Fachgruppe Methoden der DGPuK, Zürich.

Wettstein, M., Wirth, W., Reichel, K., & Kühne, R. (2012, September). *Zum Problem der Mehrfachcodierung: Sind drei wirklich genug?: Eine systematische Fehleranalyse*. Jahrestagung der Fachgruppe Methoden der DGPuK, Zürich.

Wettstein, M., Reichel, K., Kühne, R., & Wirth, W. (2012, April). *IN-TOUCH: Ein neues Werkzeug zur Prüfung und Bewertung von Codiereraktivitäten bei der Inhaltsanalyse*. Jahrestagung der Schweizerischen Gesellschaft für Kommunikationswissenschaft und Medienforschung, Neuchâtel.

Wirth, W., Kühne, R., Reichel, K., & Wettstein, M. (2011, September). *Äquivalenzprüfung als Standard in international vergleichenden Inhaltsanalysen*. Jahrestagung der Fachgruppe Methoden der DGPuK, Hannover.

Wettstein, M. (2010, September). *Term-Mapping zur komparativen Analyse öffentlicher Debatten: Eine Anwendung der Smallest Space Analysis für Inhaltsanalysen*. Jahrestagung der Fachgruppe Methoden der DGPuK, Wien.

Reichel, K., Wirth, W., Wettstein, M., & Kühne, R. (2012, September). *Die Rolle von Persönlichkeitsmerkmalen für die Qualität und Quantität des Codierverhaltens in der Inhaltsanalyse*. Jahrestagung der Fachgruppe Methoden der DGPuK, Zürich.

Eingeladene Präsentationen:

Wirth, W., & Wettstein, M. (2014, November). *Die deliberative Qualität von Leserkommentaren in Onlineforen Schweizer Tageszeitungen.* Gabriele Melischek & Josef Seethaler. Konferenz zum 20jährigen Jubiläum des Instituts für vergleichende Medien- und Kommunikationsforschung, Wien, A.

Wettstein, M. (2014, October). Populismus in der Mediengesellschaft: Untersuchung der "unheiligen Allianz" zwischen Medien und populistischen Akteuren. GV Struthonia. Wissenschaftlicher Allgemeiner Convent, Stans, CH.

Wettstein, M. (2014, October). *"Best of Both Worlds": Die halbautomatische Inhaltsanalyse.* Institut für Computerlinguistik, UZH. Master- und Doktorandenkolloquium des Instituts für Computerlinguistik, Universität Zürich, Zürich, CH.

Schwerpunkte in der Lehre:

- Methodenausbildung (Statistik I und II)
- Massenmedien und soziale Werte: Nutzung, Rezeption und Wirkung
- Kumulation und Konsonanz in der Medienwirkungsforschung
- Online Kommentare

Medienkontakte

03.08.2015	20 Minuten Online: Mops-Setire treibt Hundehalter zur Weissglut. http://www.20min.ch/schweiz/news/story/30852874
30.10.2014	20 Minuten People: Deshalb lieben wir Trash-TV. http://www.20min.ch/entertainment/tv/story/Deshalb-lieben-wir-Trash-TV-23804975
29.10.2014	Radio1-Talk of the town: Wie viel Privates darf auf Twitter veröffentlicht werden?
17.07.2014	SRF1 Rendez-Vouz: Online-Kommentare: Grosse Herausforderung für die Redaktionen http://www.srf.ch/player/radio/rendez-vous/audio/online-kommentare-grosse-herausforderung-fuer-die-redaktionen
22.02.2013	SRF2 Kultur: Amazon am Pranger. Dynamik von Shitstorms: http://www.srf.ch/kultur/gesellschaft-religion/amazon-am-pranger
20.02.2013	SRF2 Kultur: Online-Kommentare als Meinungsmacher: http://www.srf.ch/kultur/gesellschaft-religion/online-kommentare-als-meinungsmacher
08.08.2012	DRS3 Digital: Kommentare: Gepöbel, Selbstdarstellung - und fundierte Meinungen: http://www.srf.ch/wissen/digital/kommentare-gepoebel-selbstdarstellung-und-fundierte-meinungen